全国计算机技术与软件专业技术资格（水平）考试参考用书

数据库系统工程师考试科目2：

数据库系统设计与管理
——考点解析及模拟训练

全国计算机技术与软件专业技术资格（水平）考试办公室推荐

冯建华 主编



清华大学出版社

全国计算机技术与软件专业技术资格（水平）考试参考用书

数据库系统工程师考试科目2：数据库系统设计与管理——考点解析及模拟训练

冯建华　主编

清华大学出版社
北　京

内 容 简 介

本书是全国计算机技术与软件专业技术资格（水平）考试办公室推荐使用的参考用书，书中内容涵盖了数据库系统工程师考试大纲中科目 2（数据库系统设计与管理）的所有知识点，全书的主要内容有：数据库设计、数据库应用系统设计、数据库应用系统实施、数据库系统的运行和管理、SQL、网络环境下的数据库、数据库的安全性、数据库发展趋势与新技术。书中重要章节都包含考点提炼、难点解析、典型例题以及相应的习题和参考答案，有效地帮助考生进行考前复习和训练。

全书适合参加全国计算机技术与软件专业技术资格（水平）考试的考生备考使用，同时也可作为学习数据库系统设计与管理的自学用书。

图书在版编目（CIP）数据

数据库系统工程师考试科目 2：数据库系统设计与管理——考点解析及模拟训练 / 冯建华主编．—北京：清华大学出版社，2007.5
（全国计算机技术与软件专业技术资格（水平）考试参考用书）
ISBN 978-7-302-14838-8

Ⅰ．数…　Ⅱ．冯…　Ⅲ．数据库系统－工程技术人员－资格考核－自学参考资料
Ⅳ．TP311.13

中国版本图书馆 CIP 数据核字（2007）第 034347 号

责任编辑：薛　阳
责任校对：张　剑
责任印制：
出版发行：清华大学出版社　　**地　　址：**北京清华大学学研大厦 A 座
http://www.tup.com.cn　　**邮　　编：**100084
c-service@tup.tsinghua.edu.cn
社 总 机：010-62770175　　**邮购热线：**010-62786544
投稿咨询：010-62772015　　**客户服务：**010-62776969
印 刷 者：
装 订 者：
经　　销：全国新华书店
开　　本：185×230　**印张：**23.25　**防伪页：**1　**字数：**511 千字
版　　次：2007 年 5 月第 1 版　**印次：**2007 年 5 月第 1 次印刷
印　　数：1～ 000
定　　价：　　元

本书如存在文字不清、漏印、缺页、倒页、脱页等印装质量问题，请与清华大学出版社出版部联系调换。
联系电话：010-62770177 转 3103　产品编号：017441-01

前　言

全国计算机技术与软件专业技术资格（水平）考试是国家级的专业认定考试，分为计算机软件、计算机网络、计算机应用技术、信息系统、信息服务等五个专业类别。每个专业按级别层次划分为初级资格、中级资格、高级资格并有相应的资格名称。数据库系统工程师属于信息系统专业，中级资格。

本书是全国计算机技术与软件专业技术资格（水平）考试办公室推荐使用的参考用书。书中内容涵盖了数据库系统工程师考试大纲中考试科目 2：数据库系统设计与管理的全部内容。本书共分为 8 章。具体内容如下。

第 1 章是数据库设计。主要内容包括理解系统需求说明、系统开发的准备、设计系统功能、数据库设计、编写外部设计文档、设计评审等。

第 2 章是数据库应用系统设计。主要内容包括设计数据库应用系统结构、设计输入输出、设计物理数据、设计安全体系、应用程序开发、编写应用系统设计文档、设计评审等。

第 3 章是数据库应用系统实施。主要内容包括整个系统的配置与管理、常用数据库管理系统的应用、数据库应用系统的安装、数据库应用系统的测试、培训与用户支持等内容。

第 4 章是数据库系统的运行和管理。主要内容包括数据库系统的运行计划、数据库系统的运行和维护、数据库的管理、性能调整、用户支持等。

第 5 章是 SQL。主要内容包括数据库语言、SQL 概述、数据库定义、数据操作、完整性控制与安全机制、创建触发器、SQL 作用方式等。

第 6 章是网络环境下的数据库。主要内容包括分布式数据库、网络环境下数据库系统的设计与实施等。

第 7 章是数据库的安全性。主要内容包括数据库的安全性概述、数据库的安全机制、数据库的加密、数据库安全性的管理策略、数据库的安全级别等。

第 8 章是数据库发展趋势与新技术。主要内容包括面向对象数据库、ERP 和数据库、决策支持系统的建立等。

书中重要的章节都包含考点提炼、难点分析、典型例题、相应的习题和答案，帮助考生进行考前复习和训练。

本书在人国计算机技术与软件专业技术资格（水平）考试办公室的领导下组织编写。本书由冯建华主编，参与本书编写的还有：陶聪、周劲松、王道平、幺耀宗、刘华、李成都等，在此表示衷心的感谢！

由于时间仓促加之作者水平有限，书中不足之处的所难免，欢迎读者批评指正！

编　者

2007 年 1 月

目　　录

第 1 章　数据库设计

本章提示

数据库设计是建立数据库及其应用系统的第一步，是开发信息系统最重要的一部分。数据库设计中的一个核心问题，就是如何设计一个能够满足用户当前与可预见的未来的各项应用要求、性能良好的数据库。数据库设计是否合理会极大影响系统的使用性能。数据库设计就是从用户的数据需求、处理要求及建立数据库的环境条件（软、硬件特性以及其他限制）出发，运用数据库的理论知识，把给定的应用环境（现实世界）存在的数据加以合理地组织起来，逐步抽象成已经选定的某个数据库管理系统能够定义和描述的具体的数据结构，构造性能最优的数据库模式，建立数据库及其应用系统，使之能够有效地存取数据，满足各种用户的应用需求。具体地说，数据库设计包括了解用户需求、确定系统范围，选择开发方法，准备开发环境，制定开发计划，设计各子系统的功能和接口，设计安全性策略、需求和实现方法，制定详细的工作流和数据流，设计数据模型、物理结构，并编写外部设计文档。本章根据大纲的要求，全面介绍了用户需求分析、系统功能设计、数据库设计、编写外部设计文档等方面的主要知识点。在详细的典型例题分析之后，还给出了适量的练习题，以帮助读者加深对这些内容的理解和掌握。

如图 1-1 所示是本章的知识框图。

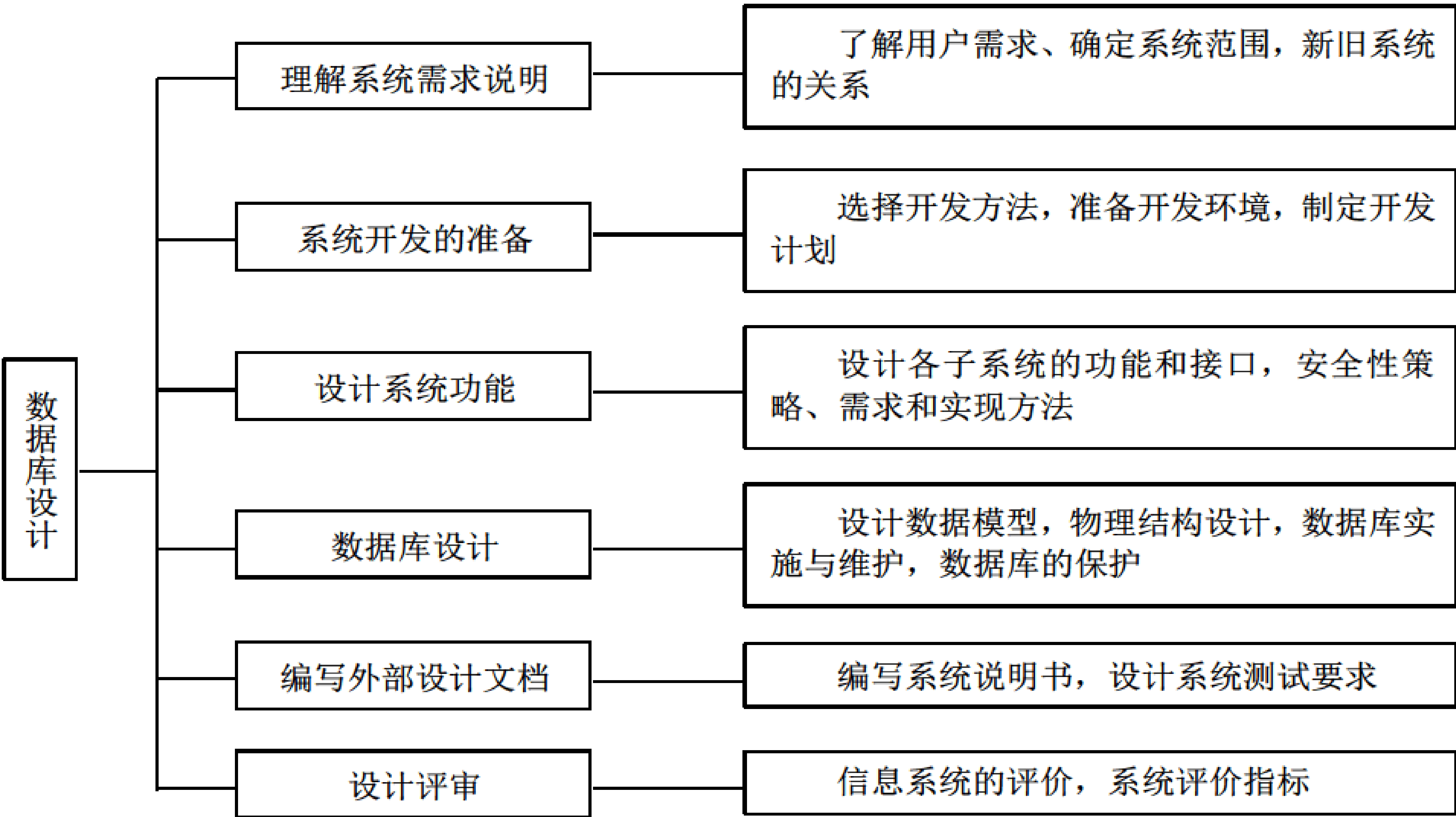


图 1-1　数据库设计知识框图

1.1　理解系统需求说明

理解系统需求是数据库设计的第一步。理解系统需求包括了解用户需求、确定系统范围，确定应用系统数据库的各种关系，现有环境与新系统环境的关系，还要确定新系统中的数据项、数据字典、数据流。如图 1-2 所示是本节的知识框图。

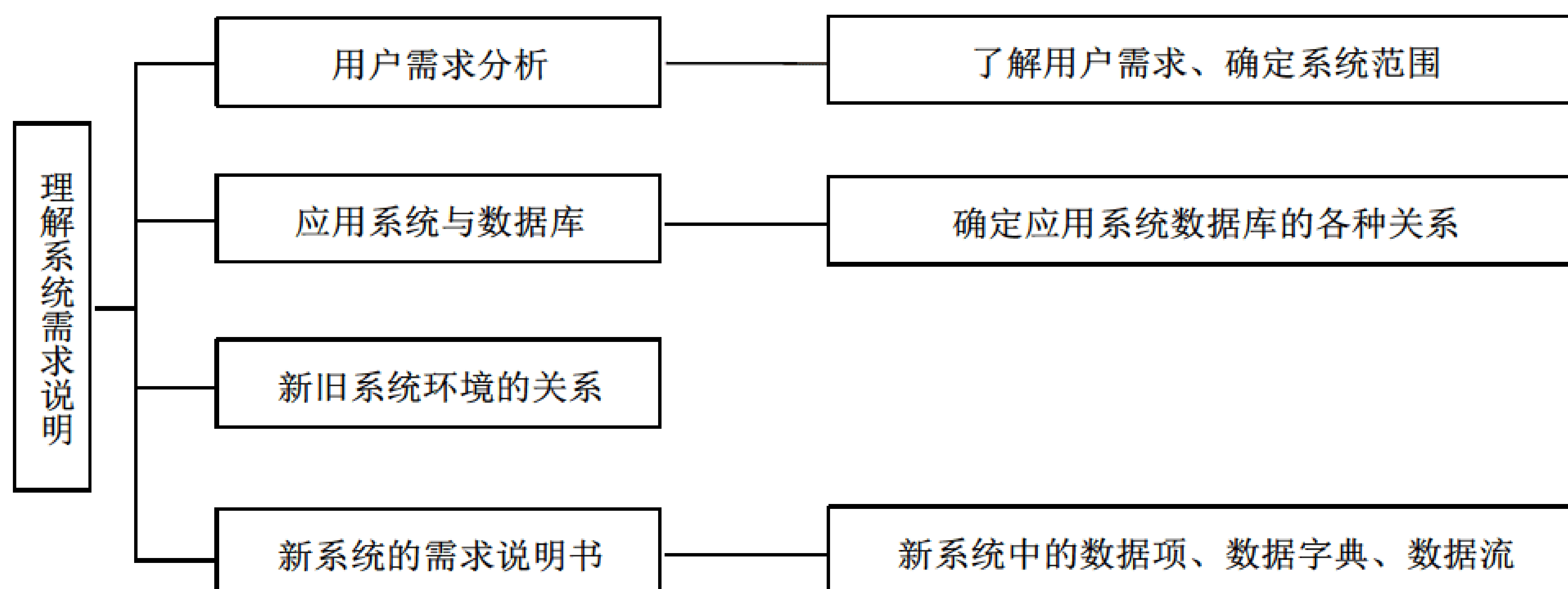


图 1-2　理解系统需求说明知识框图

1．知识点提炼

（1）了解用户需求、确定系统范围

了解用户需求、确定系统范围，即系统需求分析，它是在项目确定之后，用户和设计人员共同对数据库应用系统所要涉及的内容（数据）和功能（行为）的整理和描述，是从用户的角度出发来认识系统的。需求分析是后续开发的基础，以后的逻辑设计、物理设计以及应用程序的设计都会以此为依据。如果这一阶段的工作没有做好，势必会为以后的工作带来困难，要再重新回过头来作需求分析，会影响整个项目的工期，在人力、物力等方面造成浪费。

需求分析的基本步骤和基本要求如下所述。

① 了解将要在其中部署系统的组织（以下简称目标组织）的结构及机制；

② 了解目标组织中当前存在的问题并确定改进的可能性；

③ 确保客户、最终用户和开发人员就目标组织达成共识；

④ 导出支持目标组织所需的业务需求。

需求分析阶段的任务是：对现实世界要处理的对象（组织、部门、企业等）进行详细调查，在了解现行系统的概况，确定新系统功能的过程中，收集支持系统目标的基础数据及处理方法。需求分析是在用户调查的基础上，通过分析，逐步明确用户对系统的需求，包括数据需求和围绕这些数据的业务处理需求，以及对数据安全性和完整性方面的要求。

在需求分析的过程中，首先应当确定系统范围。在绝大多数情况下，用户并非计算机专业人员，对计算机所能处理的功能并不很了解，用户总希望所开发的系统能够尽可能多地实现他们想要的功能，而有些是目前所不可能实现的；其次，企业或部门目前可能已经有现存的系统在运行，但目前不能满足用户的要求，在新的系统中，应该继承现有系统中的数据，也可能现存系统会作为新系统中的一部分继续运行，这些都必须明确；再次，用户的应用需求，随着企业的发展，对一些可预见的需求也应当加以考虑，使新系统能够有一定的灵活性和可扩充性，适应将来的部分要求，而不仅仅是满足当前的应用需求。

需求分析阶段以调查和分析为主要手段，需要获取的主要内容有信息要求和处理要求。信息要求为用户需要在系统中保存哪些信息，由这些保存的信息要得到什么样的信息，这些信息以及信息间应当满足的完整性要求；处理要求为用户在系统中要实现什么样的操作功能，对保存信息的处理过程和方式，各种操作处理的频度、响应时间要求、处理方式等，以及处理过程中的安全性要求和完整性要求。

了解用户需求的方法就是调查。可以采取开调查会、跟班作业、查阅文献、书面填表、交流询问等方式，对用户的信息需求进行收集。收集的内容主要包括数据、业务处理的过程和依据、处理的时间和频度等。

需求分析中注意的关键问题是新系统的性能指标、目标；系统性能需求清单；业务处理和业务规则清单；当前系统业务处理文档，或者是旧版本的文档。具体如下所述：数据库中存储的数据类型？所有的数据是否都是内部数据？什么数据是外部数据？数据主要被谁使用？是否存在用户级别？数据的保存时间是多长？使用什么方法来查询和修改数据？存在哪些人工处理过程？用户当前所负责的工作是什么？用户通常与谁交互？应该给用户提供什么样的服务和产品？在数据被存档和删除前，数据能保存多长时间？为什么要限制某些用户对数据的修改？数据是字符型、数字型还是日期和时间型？信息是否必须唯一？如果是，为什么？数据的存在是否与其他数据有关？数据是否被其他数据引用？

了解用户需求后，系统设计人员要从技术的角度分析需求可行性，在允许的成本、性能要求下，分析每项需求实施的可行性，明确与每项需求实现相联系的风险，包括与其他需求的冲突，对外界因素的依赖和技术障碍。还需要确定需求的优先级别，应用分析方法来确定使用实例、产品特性或单项需求实现的优先级别。以优先级为基础确定产品版本将包括哪些特性或哪类需求。当允许需求变更时，在特定的版本中加入每一项变更，并在那个版本计划中做出需要的变更。

（2）确定应用系统数据库的各种关系

数据库应用系统就是使用数据库的各种系统（或者说各种应用程序）。确定应用系统和数据库的各种关系，才能保证应用系统的各个部分能够结合起来有效的运行。所以说数据库设计应该和应用系统的设计结合起来进行。

（3）现有环境与新系统环境的关系

现有数据库的二次设计产生的原因主要有如下几点：最终用户对系统性能不满意；公司规模扩大；公司希望提高工作效率；数据库管理员认为数据库或应用软件的安全性不能满足需求；当前系统没有问题，管理人员希望做一些不同的事情；公司希望通过自动化降低成本；管理人员有了新的财政预算等。

现有数据库的二次设计时需求分析应当注意的问题如下：当前的业务处理是什么？当前的数据如何存储？系统的不足和缺陷是什么？当前使用的硬件设备和软件环境是什么？最终用户是谁？当前系统性能如何？现有缺陷可以通过什么方法改进？新的业务处理包括哪些内容？现有的系统需要在哪些方面进行改善？根据新的业务处理，应用软件需要作哪些修改？是否涉及新数据？新数据与现有数据的相互关系？如何访问新数据？根据新数据以及数据的相互关系，数据结构需要进行哪些修改？是否会有新的用户？所进行的修改是否可以提高系统的性能？等等。

2．难点分析

需求分析阶段的成果是系统需求说明书，主要包括数据流图、数据字典、各种说明性表格、统计输出表、系统功能结构图等。系统需求说明书是以后设计、开发、测试和验收等过程的重要依据。考生应理解和掌握数据字典和数据流图的基本概念和原理，以及它们之间的关系。

数据字典是对用户信息要求的整理和描述。信息需求即定义未来信息系统用到的所有信息，用户将向数据库中输入什么信息，从数据库中要得到什么信息，各类信息的内容和结构，信息之间的联系等。数据字典通常包括数据项、数据结构、数据流、数据存储和处理过程 5 个部分。数据项：数据项是数据的最小单位，对数据项的描述一般包括项名、含义说明、别名、类型、长度、取值范围及该项与其他项的逻辑关系，常以表格的形式给出。数据结构：数据结构是若干数据项有意义的集合，通常代表某一具体的事物，包括数据结构名、含义、组成成分等。数据流：数据流可以是数据项，也可以是数据结构，表示某一次处理的输入/输出数据，包括数据流名、说明、数据来源、数据去向、及需要的数据项或数据结构。数据存储：加工中需要存储的数据，包括数据存储名、说明、输入数据流、输出数据流、组成成分、数据量、存取方式、存取频度等。处理过程：加工处理过程定义和说明。包括处理名称，输入数据、输出数据、数据存储、响应时间等。

数据流图或称数据流程图是一种便于用户理解、分析系统数据流程的图形工具。对用户处理要求的描述采用数据流图的形式，即对数据采取什么样的加工方式和操作，得到用户需要的结果，通常是对业务处理过程的描述。

数据流图描述了系统的分解，但没有对图中各成分进行说明。数据字典就是为数据流图中的每个数据流、文件、加工，以及组成数据流或文件的数据项做出说明。其中对加工的描述称为“小说明”，也可以称为“加工逻辑说明”。

3．典型例题

【例题 1-1】 阅读以下有关信息系统需求分析的叙述，回答问题 1 到问题 4。

需求分析的任务是通过详细调查现实世界要处理的对象（组织、部门、企业等），充分了解原系统（手工系统或计算机系统）工作概况，明确用户的各种需求，然后在此基础上确定新系统的功能。新系统必须充分考虑今后可能的扩充和改变，不能仅仅按当前应用需求来设计数据库。

【问题 1】 对于一般的信息系统，需求分析的重点是什么？

【解析】 需求分析的重点是调查、收集与分析用户在数据管理中的信息要求、处理要求、安全性与完整性要求。信息要求是指用户需要从数据库中获得信息的内容与性质，由用户的信息要求可以导出数据要求，即在数据库中需要存储哪些数据。处理要求是指用户要求完成什么处理功能，对处理的响应时间有什么要求，处理方式是批处理还是联机处理。新系统的功能必须能够满足用户的信息要求、处理要求、安全性与完整性要求。

【问题 2】 为什么说确定用户的最终需求是一件及其困难的事，结合自己的项目经验，谈谈确定需求的具体难点，是如何解决这些问题的？

【解析】 确定用户的最终需求其实是一件很困难的事，这是因为一方面用户缺少计算机知识，开始时无法确定计算机究竟能为自己做什么，不能做什么，因此无法一下子准确地表达自己的需求，他们所提出的需求往往不断地变化。另一方面设计人员缺少用户的专业知识，不易理解用户的真正需求，甚至误解用户的需求。此外新的硬件、软件技术的出现也会使用户需求发生变化。因此设计人员必须与用户不断深入地进行交流，才能逐步得以确定用户的实际需求。

【问题 3】 结合自己的项目经验，常用的调查方法有哪些？

【解析】

① 跟班作业。通过亲身参加业务工作来了解业务活动的情况，这种方法可以比较准确地理解用户的需求，但比较耗费时间。

② 开调查会。通过与用户座谈来了解业务活动情况及用户需求，座谈时，参加者之间可以相互启发。

③ 请专人介绍。

④ 询问。对某些调查中的问题，可以找专人询问。

⑤ 设计调查表请用户填写。如果调查表设计得合理，这种方法是很有效，也很易于被用户接受的。

⑥ 查阅记录。即查阅与原系统有关的数据记录，包括原始单据、账簿、报表等。

【问题 4】 需求分析需要完成哪些工作？

【解析】

① 首先调查组织机构情况。包括了解该组织的部门组成情况，各部门的职能等，为分析信息流程做准备。

② 然后调查各部门的业务活动情况。包括了解各个部门输入和使用什么数据，如何加工处理这些数据，输出什么信息，输出到什么部门，输出结果的格式是什么。

③ 协助用户明确对新系统的各种要求。包括信息要求、处理要求、完全性与完整性要求。

④ 确定新系统的边界。确定哪些功能由计算机完成或将来准备让计算机完成，哪些活动由人工完成。由计算机完成的功能就是新系统应该实现的功能。

【例题 1-2】 阅读以下有关信息系统需求分析的叙述，回答问题。

在设计开发某省的一个户籍管理系统时，数据库设计人员与户籍管理处的相关人员进行了沟通，了解到如下的情况：该省所有人的户籍信息都可以通过该系统查询到。户籍信息包括了户籍管理所需要的所有信息，如姓名、住址、性别、出生日期、身份证号码、照片、户主、与户主的关系等内容。

【问题】 根据以上描述提出需求分析阶段中需要了解的更多信息（不能多于 5 项，以提问的方式，选择最重要的项目）。

【解析】 需求分析阶段中需要了解的更多信息还有如下几个方面。

① 系统的最终用和和客户是谁，如何访问数据？

② 户籍数据的来源？

③ 系统将要达到的安全保密的程度如何？

④ 是否有与其他系统如身份证系统的接口？

⑤ 已故人员户籍数据是否有特殊的保存和访问方式？

【例题 1-3】 为什么说面向对象的需求分析较传统需求分析更有优势？结合自己的项目经验谈谈。

【解析】 软件的需求分析的主要目的是：通过与用户广泛的交流得出所要完成的目标系统必须具备哪些功能，应该为用户完成些什么工作，即确定“目标系统必须做什么?”。需求分析相当于从用户到软件工程人员之间架设了一道桥梁，软件工程人员通过需求分析得到用户的需求，成为软件编制所实现的目标。需求分析的好坏直接关系到软件的成功与否，是软件生命周期中的关键一环。

一般来说，用户对计算机技术了解并不多，计算机工程人员又对用户的问题不很了解，这就阻碍了用户与计算机工程人员之间的交流，使计算机工程人员不能很好地理解问题域，用户又对目标系统存在好多不清楚的地方。传统的数据流分析法，功能分析法等对这个问题并不能有效地解决。面向对象方法的出现，正好为此问题提供了一个较好的解决方案。因为人类自然地趋向于用“对象”的观点或“方法”来认识问题、分析问题以及解决问题，用基于“对象”的概念模型来建立问题域模型，自然成为系统分析员与用户交流的有效工具。

用面向对象的方法进行需求分析，其根本要点在于利用“对象”的概念模型建立一个针对于问题域的模型，用户和软件工程师通过该模型进行交流。通过在这么一个基于“对象”的问题域模型的基础上形成需求规格说明书。

【例题 1-4】 阅读以下某项目数据字典的叙述，按照问题 1 和问题 2 的提示，完成该

数据字典的定义工作。

数据字典是关于数据库中数据的描述，即元数据，而不是数据本身。数据本身将存放在物理数据库中，由数据库管理系统管理。数据字典有助于这些数据的进一步管理和控制，为设计人员和数据库管理员在数据库设计、实现和运行阶段控制有关数据提供依据。

在某学校选课系统中，要求根据各学期所开设的课程（教学任务），由不同的院（系）教学秘书为本院（系）各年级的学生选择必修课，由学生自主选择选修课，在课程学完之后，由教师进行成绩的录入。另外，教学任务应来自于各院（系）的教学计划。

【问题 1】 该系统涉及很多数据项，其中“学号”数据项可以如下描述：“学号”是每一个学生的唯一标志，一共 7 位，前 3 位标志年级号，中间 2 位标志班号，末尾 2 位按顺序编号。请按照 GB 标准写出该数据项的定义。

【解析】

数据项：　学号
含义说明：唯一标识每个学生
别名：　　学生编号
类型：　　字符型
长度：　　7
取值范围：0000000 至 9999999
取值含义：前 3 位标志年级号，中间 2 位标志班号，末尾 2 位按顺序编号

【问题 2】 “学生”是该学校选课系统中的一个核心数据结构，请按照 GB 标准写出该数据结构的定义。

【解析】

数据结构：　学生
含义说明：　定义了一个学生的有关个人信息
组成：　　　学号，姓名，系别，年级，年龄

【例题 1-5】 阅读以下某数据需求分析的叙述，回答问题。

某公司在进行项目管理时有如下的规定：项目需要技术人员支持时首先由销售人员提出申请，商务人员对销售人员的技术支持申请进行审核，检查客户是否按合同付款，如果已经按合同付款，则通过审核并提交给技术部经理进行技术审核，技术部经理接收到提交的申请后如果认为技术所需要的技术服务在合同条款内则通过审核，通过后提交到销售人员的部门经理进行综合审核，通过后则提交给技术部经理，由技术部经理指派相应的技术人员提供服务。在每一步，相关人员（商务、技术部经理、部门经理）没有通过技术支持申请，则该技术服务申请中止。

【问题】 按以上的描述画出数据流图 DFD。

【解析】 数据流图如图 1-3 所示。

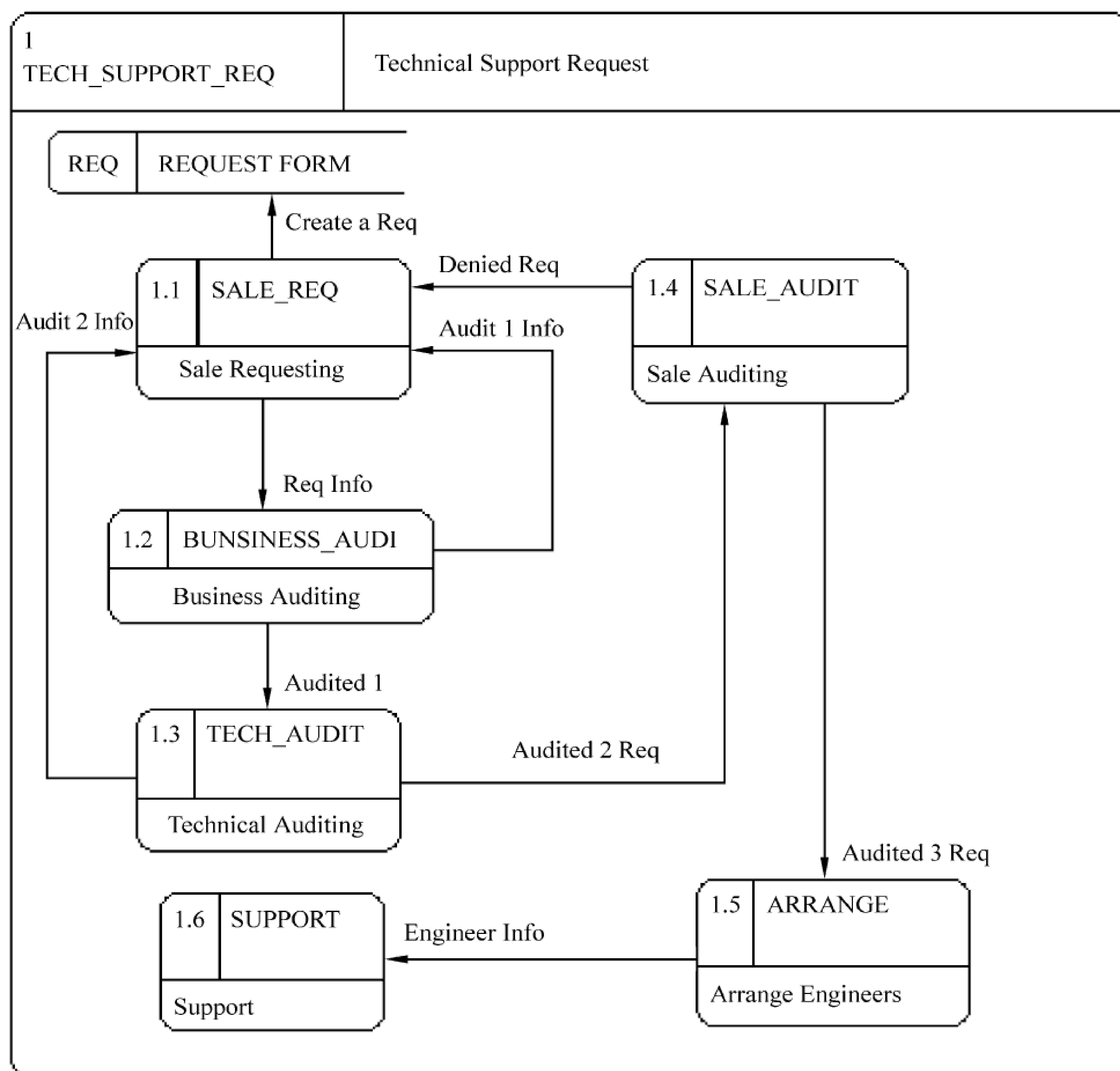


图 1-3 数据流图 DFD

1.2 系统开发的准备

系统开发前期要做一些准备工作，如在多种可行的开发方法中综合考虑各种因素选择最优的开发方法，准备开发系统所需要的软硬件环境，制定详细的开发计划以督促和保证系统能够保质保量地如期完成。如图 1-4 所示是本节的知识框图。

1．知识点提炼

（1）选择开发方法

常用的软件开发方法有 3 种：结构化方法、面向对象方法和原型法。

结构化方法是结构化分析和结构化设计的总称。结构化方法的特点是有一套严格的开发程序，各开发阶段都要求有完整的文档记录。

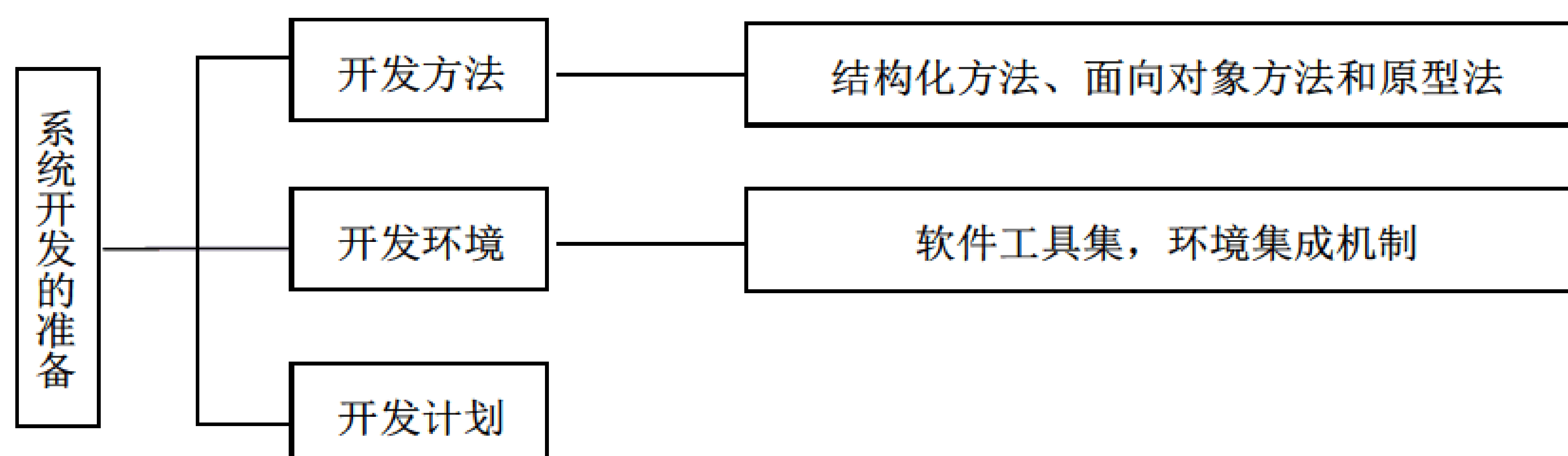


图 1-4　系统开发的准备知识框图

面向对象方法也包含面向对象的分析和面向对象的设计。面向对象方法的特点是从现实世界中客观存在的事物出发来构造软件系统。软件系统使用的业务范围称做软件的问题领域，把问题领域中事物的特征抽象地描述成类，由类建立的对象做为系统的基本构成单位，它们的内部属性与服务描述了客观存在的事物的静态特征和动态特征。对象类之间的继承关系、聚集关系、消息和关联反映了问题域中事物之间实际存在的各种关系。

原型化方法的特点是：先获得一组基本的需求后，快速地加以“实现”；随着用户或开发人员对系统理解的加深而不断地对这些需求进行补充和细化，系统的定义是在逐步发展的过程中进行的，而不是一开始就预见一切。原型化方法是确定需求的策略，对用户的需求进行抽取、描述和求精。它快速地迭代并建立最终系统的工作模型，它对问题的定义采用启发的方式，并由用户做出响应。原型化方法特别适用于最初不能严格定义用户需求或需求经常发生变化的系统。

（2）准备开发环境，制定开发计划

在具体研究需求分析之前，先了解一下软件工程这个概念。软件工程分为 3 个层次：过程层、方法层、工具层。在最基础的过程层，最重要的就是一组被称为关键过程区域（KPAs）的框架（KPA 的概念在讨论 CMM 的书中有详细的概念说明）。关键过程区域构成了软件项目管理控制的基础，并且确立了上下文各区域的关系，其中规定了技术方法的采用，工程产品的模型、文档、数据、报告、表格等的产生，里程碑的建立，质量的保证及变化的适当管理。方法层主要是过程在技术上的实现，它解决的问题是如何做。软件工程方法涵盖了一系列的任务：需求分析、设计、编程、测试、维护，同时它还包括了一组基本原则，控制了每一个的关键过程区域。工具层就很好理解了，它对过程层和方法层提供了自动和半自动的支持。这些辅助工具就称为 CASE。

2．难点分析

软件开发环境是指支持软件产品开发的软件系统，它由软件工具集和环境集成机制构成。

用来辅助软件开发、运行、维护、管理、支持等过程中的活动的软件称为软件工具，工具集应包括支持软件开发相关过程、活动、任务的软件工具，以对软件开发提供全面的支持。软件工具按照软件过程的活动可以划分为支持软件开发过程的工具、支持软件维护

过程的工具、支持软件管理过程和支持过程的工具等。用户应该根据费用、功能、易用性、稳健性、硬件要求和性能、服务支持等标准来综合选择软件开发工具。

软件开发环境是支持软件产品开发的软件系统。它由软件工具集和环境集成机制构成，前者用来支持软件开发的相关过程、活动；后者为工具集成和软件开发、维护和管理提供统一的支持，它通常包括数据集成、控制集成和界面集成。通过环境集成机制，各工具用统一的数据接口规范存储或访问环境信息库；各工具采用统一的界面形式，保证各工具界面的一致性，同时为各工具或开发活动之间的通信、切换、调度和协同工作提供支持。在软件开发环境中进行软件开发，可以使用环境中提供的各种工具，同时在环境信息库的支持下，一个工具所产生的结果信息可以被其他工具利用，使得软件开发的各项活动得到连续的支持。

3．典型例题

【例题 1-6】 目前大型数据库管理系统有多种，请根据自己的项目经验，从总体上评价目前主流的大型数据库。

【解析】 这里主要引用 SQL Server、DB2、Oracle、Sybase 4 种数据库，Informix 也是非常不错的数据库，现已经与 DB2 合为一体。无论是从性能、技术、安全上考虑，Oracle 都是不错的选择，适用于各种级别的应用。DB2 紧跟其后，但是市场份额方面，主要在大型机上占绝对优势。性价比方面 SQL Server 还不错，可用性和易用性是最好的，但是安全性不够理想。在考虑多平台时，Sybase 是 SQL Server 的很好的替代品。

开源数据库中，MySQL 和 Postgre SQL 都非常优秀，虽然在事务处理方面，各自都有些缺陷，但是使用成本却是最低的。

【例题 1-7】 就开放性、可伸缩性、并行性、性能、易用性、使用风险等方面，列举采购微软公司 SQL Server 数据库时的注意点。

【解析】 SQL Server 只能在 Windows 上运行，没有丝毫的开放性，操作系统的系统的稳定对数据库是十分重要的；Windows 9X 系列产品是偏重于桌面应用，NT Server 只适合中小型企业；而且 Windows 平台的可靠性、安全性和伸缩性是非常有限的；它不像 UNIX 那样久经考验，尤其是在处理大数据量的关键业务时；并行实施和共存模型并不成熟；很难处理日益增多的用户数和数据卷，伸缩性有限；SQL Server 还没有获得任何安全证书，安全性不够理想。相对其他大型商业数据库，SQL Server 在多用户并发时性能不佳；操作简单，但只有图形界面；并不十分兼容早期产品；使用需要冒一定风险。

【例题 1-8】 就开放性、可伸缩性、并行性、性能、易用性、使用风险等方面，列举采购 Oracle 数据库时的注意点。

【解析】 Oracle 能在所有主流平台上运行；完全支持所有的工业标准；采用完全开放策略；可以使客户选择最适合的解决方案；对开发商全力支持；特别是 Oracle 并行服务器通过使一组结点共享同一簇中的工作来扩展 Windows NT 的能力，提供高可用性和高伸缩性的簇的解决方案；如果 Windows NT 不能满足需要，用户可以把数据库移到 UNIX 中；

Oracle 的并行服务器对各种 UNIX 平台的集群机制都有着相当高的集成度；获得最高认证级别的 ISO 标准认证；性能非常好，在开放平台下的 TPC-D 和 TPC-C 测试记录都很好；同时提供 GUI 和命令行，在 Windows NT 和 UNIX 下操作相同，但是操作还是较复杂；由于 Oracle 公司长时间的数据库开发经验，向下兼容性好，得到广泛的应用；风险非常小。

【例题 1-9】 就开放性、可伸缩性、并行性、性能、易用性、使用风险等方面，列举采购 Sybase ASE 数据库时的注意点。

【解析】 Sybase ASE 能在所有主流平台上运行；但由于早期 Sybase 与 OS 集成度不高，因此 11.9.2 以下版本需要较多 OS 和 DB 级补丁；在多平台的混合环境中，会有一定问题；12.0 以上版本得到了一定的改进；虽然有 DB Switch 来支持其并行服务器，但由于 DB Switch 在技术层面还未成熟，且只支持版本 12.5 以上的 ASE Server，因为 DB Switch 技术需要一台服务器充当 Switch，从而在硬件开销上会有所增加；在安全方面，获得最高认证级别的 ISO 标准认证；性能接近于 SQL Server，但在 UNIX 平台下的并发性要优与 SQL Server；操作较复杂，同时提供 GUI 和命令行；但 GUI 较差，常常无法及时更新状态，使用命令行模式时较为复杂；向下兼容，但是 ct-library 程序不益移植。

【例题 1-10】 就开放性、可伸缩性、并行性、性能、易用性、使用风险等方面，列举采购 IBM 公司 DB2 数据库时的注意点。

【解析】 DB2 能在所有主流平台上运行；最适于海量数据；DB2 在企业级的应用最为广泛；具有很好的并行性；DB2 把数据库管理扩充到了并行的、多节点的环境。数据库分区是数据库的一部分，包含自己的数据、索引、配置文件、和事务日志。数据库分区有时被称为节点或数据库节点；在安全方面，获得最高认证级别的 ISO 标准认证；性能较高适用于数据仓库和在线事物处理；操作简单，同时提供 GUI 和命令行，在 Windows NT 和 UNIX 下操作相同；在巨型企业得到广泛的应用，向下兼容性好；风险小。

【例题 1-11】 阅读以下关于软件开发模式方面的叙述，回答问题 1 和问题 2。

近年来，美国有名的 Rational 软件公司根据多年来软件开发的实践与理论，倡导了一种软件开发过程的“瑞理模式”（Rational Approach），其中强调了以下几个要点：面向对象；螺旋式上升的渐进；管理与控制；高度自动化。其中心要素则是人力、过程（方法）以及工具，从而能广泛适用于许多领域的软件产品生产与软件项目开发。

概括地说，在该模式中采用管理观点和技术观点这两种不同的观点互为补充地描述了软件开发的渐进式过程。

① 管理观点——侧重于在开发过程中对财力、策略与人员等方面的管理，按照管理观点，把软件生命周期中的项目进程区分为下列 4 个主要阶段。

- 开始阶段：分析设计思想，构思未来软件产品的原型，定义使用场合与项目范围。
- 规划阶段：具体规划项目实施所需的作业活动与投入的资源，规定软件特性，构筑设计框架。
- 构建阶段：根据已规划的软件原型、框架与作业活动，逐步去开发与构建出软件

产品，直至开发出相对完整的产品。

- ❑ 移交阶段：把已开发好的软件产品设法移交给用户使用，包括加工、交付、培训、支持、维护等直至满足合同要求和使得用户满意为止。

由上述 4 个阶段组成的一个“开发周期”，可开发出某一版本的软件产品。再统称这类新一版软件开发过程为“进化阶段”（即包括下一轮的开始、规划、构建与移交过程）。

② 技术观点——把一个软件的开发看成为一连串循环（迭代）所组成，每一循环（迭代）都包括了计划、分析、设计、编程及测试等活动，即去完成软件开发过程中一个“完整而独立的小部分”的功能需求。综合了所有循环的产出结果，可获得完整的软件产品。环绕着人力、过程（方法）和工具，技术观点与管理观点可以相互在时间进程上做出映像，即在管理观点的开始、规划、构建、移交等每一阶段中都可以按技术观点去划分成若干循环（迭代），由各个循环逐步地去分别完成软件产品的各阶段的某一表现形式（如概念原型、结构原型、结构基准、预备板、试用版、提交版 1、提交版 2 等）。

【问题 1】 在传统讨论的软件开发过程中，把软件项目开发过程描述为研究与开发时期（R&D）、生产时期和维持时期，这与瑞理模式的管理观点中的各个阶段大体上有什么对应关系？

根据自己的开发经验，在一个中等规模软件项目中，在管理观点的最初开发期中的哪一阶段所花的精力和时间最多，用什么主要措施可以减少该阶段所占的比重？（100 字以内）

【解析】

① 开始阶段加上规划阶段相当于研究开发时期；

② 构建阶段加上移交阶段相当于生产时期；

③ 进化阶段相当于维护时期；

④ 时间与精力花费最多的阶段是“构建”阶段。

采用合适的相应软件方法、工具与程序环境可减少构建阶段所占的比重。

【问题 2】 瑞理模式不把重点放在文件的制作及软件的外观表面形式上，强调的是软件产品本身及其质量和用户的满意程度。该模式结合了管理观点讨论中涉及的 5 个阶段和技术观点的螺旋式上升技术。请说明该模式特别适合于哪种类型的软件项目，为什么？（100 字以内文字）

【解析】 特别适合于需求有变动的处理和有着较高风险的软件项目部分。

因为把软件产品细分为一系列循环或迭代，每理去实现一个完整而独立的小单元，预先规划地进入原则与产出结果，遇到变动或特殊情况可及早解决，并可把变动的需求加入到将来实现的循环或迭代的计划之中。

【例题 1-12】 根据自己的项目经验，考虑选择大型分布式数据库管理系统时应从哪几个方面予以考虑？每个方面的注意点是什么？

【解析】 选择大型分布式数据库管理系统时应从以下几个方面予以考虑：

数据库管理系统的性能。包括性能评估（响应时间、数据单位时间吞吐量）、性能监控（内外存使用情况、系统输入/输出速率、SQL 语句的执行、数据库元组控制）、性能管理（参数设定与调整）、并行处理能力、是否支持多 CPU 模式的系统（SMP，Cluster，MPP）、负载的分配形式，并行处理的颗粒度、范围。

并发控制功能。对于分布式数据库管理系统，并发控制功能是必不可少的。评价并发控制的标准包括保证查询结果一致性方法、数据锁的颗粒度（数据锁的控制范围、表、页、元组等）、数据锁的升级管理功能、死锁的检测和解决方法等。

容错能力。异常情况下对数据的容错处理评价标准包括硬件的容错，有无磁盘镜像处理功能；软件的容错，有无软件方法异常情况的容错功能。

安全性控制。包括安全保密的程度（账户管理、用户权限、网络安全控制、数据约束）

程序开发的难易程度。有无计算机辅助软件工程工具 CASE——计算机辅助软件工程工具，可以帮助开发者根据软件工程的方法提供各开发阶段的维护、编码环境，便于复杂软件的开发、维护。

支持汉字处理能力。包括数据库描述语言的汉字处理能力（表名、域名、数据）和数据库开发工具对汉字的支持能力。

以上是技术方面的一些考虑，非技术方面还有软件厂家的服务支持、经济实用性等。

【例题 1-13】 阅读以下关于软件开发环境方面的叙述，回答问题。

某企业的管理系统中，生产系统使用的环境如下所示。

主机：SUN 10K，12 个 CPU，12GB RAM，5TB 磁盘。

操作系统：SUN Solaris 8。

数据库：Oracle 8.0.5。

开发环境如下所述。

主机：SUN 15K，12 个 CPU，12GB RAM，10TB 磁盘。

操作系统：SUN Solaris 9。

数据库：Oracle 8.1.6。

【问题】 请指出以上开发环境配置的 3 个问题，按重要程度排列。

【解析】 开发环境配置的 3 个主要问题如下所述。

① 开发环境中使用的数据库版本与生产环境不同。

② 开发环境中的操作系统版本与生产环境中的不同。

③ 开发环境使用的硬件平台的性能好于生产环境。

1.3　设计系统功能

一般的信息系统由应用系统和数据库两大部分构成。系统功能设计是信息系统开发过程中的一个重要阶段。系统设计主要目的就是为系统制定蓝图，在各种技术和实施方法中

权衡利弊，精心设计，合理利用各种资源，选择系统结构，设计各子系统的功能和接口；设计安全性策略、需求和实现方法；制定详细的工作流和数据流，最终形成详细设计方案。系统设计的主要内容包括总体结构设计、子系统设计、模块设计、代码设计、输出设计、输入设计、处理过程设计、数据存储设计、用户界面设计和安全控制设计等。如图 1-5 所示是本节的知识框图。

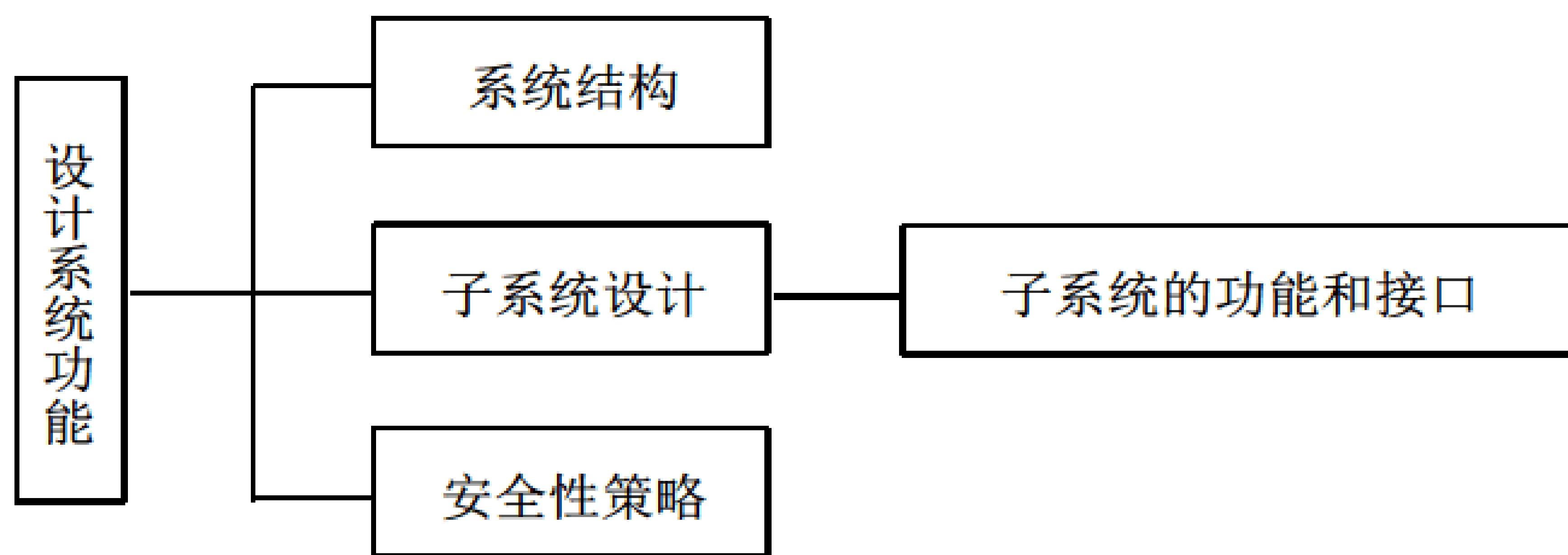


图 1-5　设计系统功能知识框图

1．知识点提炼

（1）设计各子系统的功能和接口

系统功能设计可以分为以下 3 个步骤。

① 把整个系统分解为许多基本的、具体的子系统，将子系统合理地组织起来构成整个系统，实现需求分析阶段确定的系统功能。

② 将子系统进一步细分成许多模块，这个过程称为总体结构设计（Architecture Design），又称为概要结构设计（Preliminary Design）。概要结构设计的基本任务是：将系统划分为模块，决定每个子模块要实现的功能和模块的调用关系。

③ 为各个子模块选择适当的技术手段和处理方法，该过程称为详细设计。详细设计基本任务包括代码设计、输出设计、输入设计、处理过程设计、数据存储设计、用户界面设计、安全控制设计。

（2）设计安全性策略

一个好的数据库不仅应该充分满足组织的各级管理要求，使得后继系统开发工作方便、快捷，具有系统开销小，易于管理和维护等特点，还应该满足信息安全性的要求。

2．难点分析

一般数据库软件都提供定义数据安全保密性的基本功能。系统所提供的安全保密功能一般有 8 个等级（0～7 级），4 种不同方式（只读、只写、删除、修改），而且允许用户利用这 8 个等级的 4 种方式对每一个表自由地进行定义。建立合适的安全级别和方案，是整个系统有效运行的重要保证。制定安全性策略，首先要确定系统的安全性需求，然后为系统的不同模块一一制定相应的安全性策略，并设计实现方法。

3．典型例题

【例题 1-14】阅读以下关于信息系统查询与查错设计方面的叙述，回答问题 1 和问题 2。

某物资部门的财务管理信息系统主要用于财会核算工作的全面管理，包括对物资与资金分科制登记 3 级明细账，产生凭证，总账平衡，成本核算，利润分配，总账生成，产生各阶段报表，打印明细账与凭证，对账务进行查询、修改、分析、转储等功能，采用微机网络方式运行该信息系统。在该系统分析与设计过程中，十分重视查询与资料查错功能的设计。

在查询设计中，主要提供明细账数据文件、记账凭证数据文件和财务报表文件 3 大类查询。同时提供了以下 3 种查询方式。

① 全局自动查询——这是一种批量资料的查询，方便于用户模糊查询。即根据用户所选择输入的某些查询值（用户可能仅记住资料的部分特征），自动对整个文件系统进行全面搜索，从各个数据文件中找出满足用户查询条件的全部相应记录，供用户参考选用。

② 相关文件查询——由于在财会账务中各类明细账和财务报表文件之间存在着相当密切的联系，经常需要查询在多个数据文件中具有相互关联的资料或记录。相关文件查询具有联接资料的特征，根据用户选取规定的查询条件值，把存在于多个数据文件中的相应数据组织成新的资料信息，并可以同时核查多个用户数据文件。

③ 组合条件查询——一这不同于通常的固定条件查询，在查询过程中用户可随机指定若干查询条件，由系统去生成相应的查询。

【问题 1】 组合条件查询由用户从查询某单位中选若干项查询条件，指定各查询条件之间的逻辑关系（即“与”、“或”关系），由系统自动生成用户所需的组合查询表达式，从而去检索相应的数据文件。请以 100 字以内文字简要叙述组合条件查询的主要优点。

【解析】

① 提供交互式方便查询，提高了用户查询的灵活性。

② 在一定程度上适应查询需求的变动。

③ 可适用于用户未能确切地提供查询条件具体表达式的场合。

④ 在大量不同查询条件下，可显著减少设计查询的总工作量。

【问题 2】 在财务信息系统中，防止资料录人出错是十分重要的一个任务，在该系统设计时采用了以下 3 类检查出错的功能设计。

① 资金平衡查错——根据财务中的资金来源科目和资金占用科目之间差额平衡原理设计的，通过“平衡监视程序”对各种资料录入时所产生的记账凭证内容进行平衡，把平衡的结果反馈给财会人员，由财会人员判别录入资料的正确性。

② 科目核对查错——在一个“文件控制库”内存放着供核对用的科目编号，每当财会人员输人错误的科目编号时，“科目核对校验程序”将核对“文件控制库”后提示财会人员。

③ 资金汇总核对查错——在发票、入库单、日记账等资料录入过程中，可以采用累加器把录入的每个资料记录中的资金或数量等进行累加，在自动生成凭证之前把累加获得的金额与数量等结果告诉财会人员，由财会人员根据单据人工辅助核对，以确认资料是否

正确。

请以 150 字以内文字简要说明，从明细账数据库文件角度来看，为了提高工作效率和保证数据录入可靠正确，这 3 类查错中数据存放的共同设计特征是什么？

【解析】 增设临时文件，在查错时录入的资料存入临时文件（不记入明细账）。查出资料错误时，允许进行局部性修改。在财会售货员确定资料准确后，自动地把资料记入相应的明细账。

1.4　数据库设计

数据库设计是指对于一个给定的应用环境，构造最优的数据库模式，建立数据库及其应用系统，使之能有效地存储数据，满足各种用户的需求（信息要求和处理要求）。本节主要讲述概念结构设计（设计 E-R 模型）、逻辑结构设计、物理结构设计、数据库实施与维护、数据库的保护等。如图 1-6 所示是本节的知识框图。

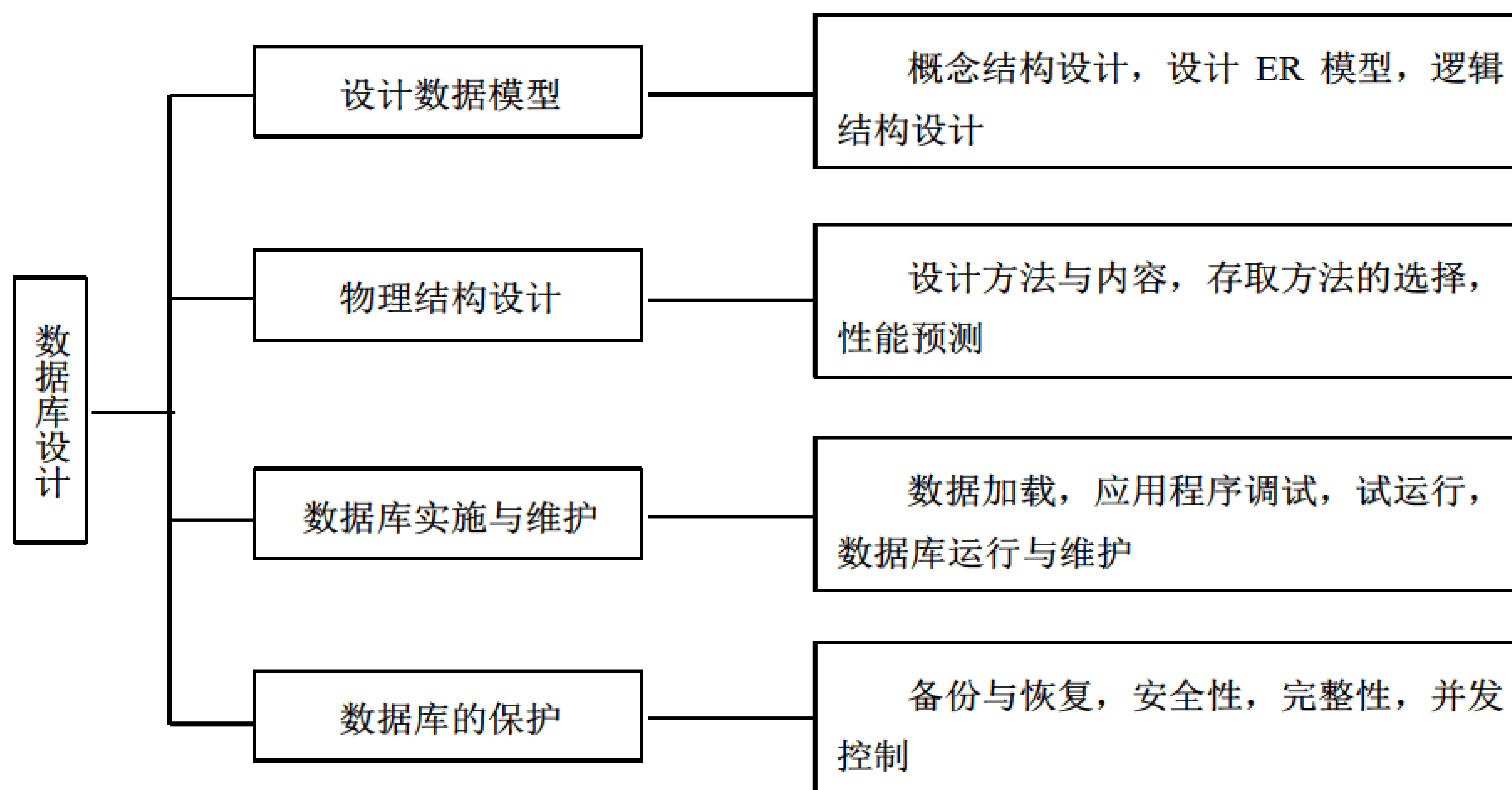


图 1-6　数据库设计知识框图

1.4.1　设计数据模型

1．知识点提炼

数据库概念结构设计阶段是在需求分析的基础上，依照需求分析中的信息要求，对用户信息加以分类、聚集和概括，建立信息模型，并依照选定的数据库管理系统软件，转换为数据的逻辑结构，再依照软硬件环境，最终实现数据的合理存储。这一过程也称为数据建模，可分为概念结构设计、逻辑结构设计和物理结构设计 3 个过程，它们是本节的重点，考生务必注意。

（1）概念结构设计（设计 E-R 模型）

概念结构设计是设计人员以用户的观点，对用户信息的抽象和描述，从认识论的角度来讲，是从现实世界到信息世界的第一次抽象，并不考虑具体的数据库管理系统。对现实世界的事物加以分类整理，理清各类信息之间的关系，描述信息处理的流程，这一过程就是概念结构设计。

概念结构设计的策略通常有以下 4 种：自顶向下，即首先定义全局概念结构的框架，然后逐步细化；自底向上，即首先定义各局部应用的概念结构，然后将它们集成起来，得到全局概念结构；逐步扩张，即首先确定核心业务的概念结构，然后以此为中心向外扩张，最终实现全局概念结构；混合策略，即将自顶向下和自底向上两种策略结合使用，首先确定全局框架，划分为若干个局部概念模型，再采取自底向上的策略实现各局部概念模型，加以合并实现全局概念模型。实际应用中这些策略并没有严格的限定，可以根据具体业务的特点选择。

目前，绝大多数 DBMS 都是基于关系模型的，其中 E-R（实体-联系模型）方法是概念结构设计的主要方法。使用 E-R 方法，无论是应用哪种策略，都需要对现实事物加以抽象认识，以 E-R 图的形式描述出来。对现实事物进行抽象认识主要有 3 种方法，即分类、聚集和概括。分类（Classification）是指对现实世界的事物，按照其具有的共同特征和行为定义一种类型。聚集（Aggregation）是指定义某一类型所具有的属性，通过在这些属性上的不同取值来区分不同的个体。概括（Generalization）是指由一种已知类型定义新的类型，通常把已知类型称为超类（Superclass），新定义的类型称为子类（Subclass），子类是超类的一个子集。按照以上 3 种抽象机制，对系统需求分析阶段所得到的数据进行分类、聚集和概括，从而确定实体、属性和联系等，完成 E-R 图的设计。

（2）逻辑结构设计（转换成 DBMS 所能接收的数据模型）

逻辑结构设计是在概念结构设计的基础上，按照选用的 DBMS，进行数据模型设计，数据模型可以是层次模型、网状模型和关系模型。目前，大多数 DBMS 都是基于关系模型的，E-R 方法是概念结构设计的主要方法，因此如何在全局 E-R 图基础上进行关系模型的逻辑结构设计成为逻辑结构设计的主要内容。

逻辑结构设计阶段主要完成以下 4 项任务：首先，确定数据模型；其次，将 E-R 图转换成为指定的数据模型；再次，确定完整性约束；最后，确定用户视图。

因为目前大多数 DBMS 都是基于关系模型的，所以要将 E-R 图转换成为关系模型。然而，E-R 图是由实体、属性和联系三要素构成，而关系模型中只有唯一的结构——关系模式。可以采取下面的方法加以转换：首先，将 E-R 图中的实体逐一转换成为一个关系模式，实体名对应关系模式的名称，实体的属性转换成关系模式的属性，实体标识符就是关系的码。其次，将联系向关系模式的转换。E-R 图中有一对一联系（1∶1）、一对多联系（1∶n）和多对多联系（m∶n）3 种联系，分别对应着不同的转换方法。各种联系具体如下所述。

① 一对一联系（1∶1）。设 A、B 为两个实体集，若 A 中的每个实体至多和 B 中的一

个实体有联系，反过来，B 中的每个实体至多和 A 中的一个实体有联系，称 A 对 B 或 B 对 A 是 1∶1 联系。需要注意的是 1∶1 联系不一定都是一一对应的关系，可能存在着无对应的情况。

② 一对多联系（1∶*n*）。如果实体集 A 中的每个实体可以和 B 中的几个实体有联系，而 B 中的每个实体至只和 A 中的一个实体有联系，则称 A 对 B 是 1∶*n* 联系。

③ 多对多联系（*m*∶*n*）。若实体集 A 中的每个实体可与和 B 中的多个实体有联系，反过来，B 中的每个实体也可以与 A 中的多个实体有联系，称 A 对 B 或 B 对 A 是 *m*∶*n* 联系。必须强调指出：有时联系也有属性，这类属性不属于任一实体只能属于联系。

由 E-R 图转换而来的关系模式会有数据冗余、更新异常存在，因此需要将关系模式规范化，具体操作步骤如下所述。

① 根据语义确定各关系模式的数据依赖。在设计的前一阶段，只是从关系及其属性来描述关系模式，并没有考虑到关系模式中的数据依赖。关系模式包含着语义，要根据关系模式所描述的逻辑语义，写出关系数据依赖。

② 根据数据依赖确定关系模式的范式。由关系的码及数据依赖，根据规范化理论，就可以确定关系模式所属的范式，判定关系模式是否符合要求，即是否达到了 3NF 或 4NF。

③ 如果关系模式不符合要求，要根据关系模式的分解算法对其进行分解，达到 3NF、BCNF 或 4NF。

④ 关系模式的评价及修正。根据规范化理论，对关系模式分解之后，就可以在理论上消除冗余和更新异常，但根据处理要求，可能还需要增加部分冗余以满足处理要求，就需要做部分关系模式的处理，分解、合并或增加冗余属性，提高存储效率和处理效率。

对关系模式进行了规范化处理之后，还需要对关系模式加以完整性约束，包括数据项的约束、表级约束及表间约束。可以参照 SQL 标准来确定不同的约束，如检查约束、主码约束和参照完整性约束等，以保证数据的正确性和完备性。

关系模式确定之后，需要分别根据数据流图和用户信息确定视图模式，将数据视图和用户视图分离开，以提高数据的安全性和独立性。

首先，根据数据流图选择处理过程使用的数据视图。数据流图是某项业务的处理，使用了部分数据，这些数据可能要跨越不同的关系模式，建立该业务的视图，可以降低应用程序的复杂性，并提高数据的独立性。其次，根据用户类别选择不同用户所使用的用户视图。对于不同的用户而言，他们只有处理整个系统的某一部分或者某几部分数据的权限，因此需要选择用户视图以保证数据的安全性。

2．难点分析

E-R 模型的设计是概念结构设计的难点，考生应特别注意。设计 E-R 模型的步骤如下所述。

首先，选择适当层次的数据流图，让该层的每一部分对应一个局部应用，实现某一项功能。其次，根据划分的各个局部应用设计分 E-R 图，又称为局部 E-R 图。最后，将局部

E-R 图合并，得到一张全局 E-R 图。合并的方法是将具有相同实体的两个或多个 E-R 图合而为一，在合成后的 E-R 图中把相同实体用一个实体表示，合成后的实体的属性是所有分 E-R 图中该实体属性的并集。并以此实体为中心，并入其他所有分 E-R 图，再把合成后的 E-R 图以分 E-R 图看待，合并剩余的分 E-R 图，直至所有的 E-R 图全部合并，得到全局 E-R 图。在合并过程中要解决分 E-R 图中相互间存在的冲突，消除分 E-R 图之间存在的信息冗余，使之成为能够被全系统所有用户共同理解和接受的统一的、精炼的全局概念模型。

分 E-R 图之间的冲突主要有以下 3 类。

① 属性冲突，即同一属性可能会存在于不同的分 E-R 图，由于设计人员不同或是出发点不同，对属性的类型、取值范围、数据单位等可能会不一致，这些属性对应的数据将来只能以一种形式在计算机中存储，这就需要在设计阶段进行统一。

② 命名冲突，即相同意义的属性，在不同的分 E-R 图上有着不同的命名，或是名称相同的属性在不同的分 E-R 图中代表着不同的意义，这些也要进行统一。

③ 结构冲突，即同一实体在不同的分 E-R 图中有不同的属性，同一对象在某一分 E-R 图中被抽象为实体，而在另一分 E-R 图中又被抽象为属性，需要统一。

合并局部 E-R 图的过程中，注意要从 3 个方面对其进行优化。一是合并实体，即两个具有 1∶1 联系或 1∶n 联系的实体，可以予以合并，使实体个数减少，有利于减少将来数据库操作过程中的连接开销。二是消除冗余属性，因为合并后的 E-R 图中的实体继承了合并前该实体在分 E-R 图中的全部属性，属性间就可能存在冗余，即某一属性可以由其他属性确定，所以要消除冗余属性。三是消除冗余联系，在合并局部 E-R 图的过程中，可能会出现实体联系的环状结构，即某一实体 A 与另一实体 B 间有直接联系，同时 A 又通过其他实体与实体 B 发生间接联系，通常直接联系可以通过间接联系所表达，可以消除直接联系。

得到全局 E-R 图，也就完成了概念结构设计。

3．典型例题

【例题 1-15】 数据库设计一般包括哪几个方面，每一个方面设计的具体内容是什么？

【解析】 数据库设计一般包括 3 个方面：概念结构设计阶段，通过对用户需求进行综合、归纳与抽象，形成一个独立于具体 DBMS 的概念模型，可以用 E-R 图表示；逻辑结构设计阶段，将概念结构转换为某个 DBMS 所支持的数据模型（例如关系模型），并对其进行优化；数据库物理设计阶段，为逻辑数据模型选取一个最适合应用环境的物理结构（包括存储结构和存取方法）。

【例题 1-16】 简述 E-R 图的构成要素。

【解析】 E-R 图有以下 3 个要素。

① 实体：用矩形表示实体，矩形内标注实体名称。

② 属性：用椭圆表示属性，椭圆内标注属性名称，并用连线与实体连接起来。

③ 实体之间的联系：用菱形表示，菱形内注明联系名称，并用连线将菱形框分别与

相关实体相连，并在连线上注明联系类型。

【例题 1-17】 根据自己的项目经验，说说为什么要创建视图（View）？视图创建的原则一般有哪些？

【解析】 简化用户所见到的数据，基于安全的考虑，从多表中抽取数据，提高查询性能，为用户提供另外一种数据组织方式。

避免创建多于两级或三级的视图，视图应该被模式所拥有，必须提供关于视图使用的文档。

【例题 1-18】 规范化理论是研究如何将一个不好的关系模式转化为好的关系模式的理论，规范化理论是围绕范式而建立的。数据库中的数据规范化的优点是减少了数据冗余，节约了存储空间，相应逻辑和物理的 I/O 次数减少，同时加快了增、删、改的速度。但是规范化的程度并非越高越好，根据自己的项目经验，你认为反规范化的目的是什么，一般包含哪些手段？

【解析】 反规范化的目的主要是为了提高数据库的性能。通常包含下面 3 种手段。

① 复制某些数据列到一些表中以便更容易地访问它们而不用进行多表的连接，这些被复制的列可以是它们自己的列或外码列。

② 预计算和派生数据的存储可以加快处理过程。

③ 撤销某些分解的实体是为避免多个连接的开销。

【例题 1-19】 简述关系的性质。

【解析】 关系的性质有如下几个方面。

① 关系中一列的各个分量具有相同的性质，即数据类型相同。

② 关系中行的顺序、列的顺序可以任意互换，不会改变关系的意义。

③ 关系中的任意两个元组不能相同。

④ 关系中的元组分量具有原子性，即每一个分量都必须是不可分的数据项。

【例题 1-20】 阅读以下概念设计的叙述，回答问题 1 和问题 2。

将需求分析得到的用户需求抽象为信息结构即概念模型的过程就是概念设计。概念结构是对现实世界的一种抽象，即对实际的人、物、事和概念进行人为处理，抽取人们关心的共同特性，忽略非本质的细节，并把这些特性用各种概念精确地加以描述。

【问题 1】 一般来说，概念设计的方法有哪些，各有什么特点？

【解析】 设计概念结构通常有以下 4 类方法。

① 自顶向下。即首先定义全局概念结构的框架，然后逐步细化。

② 自底向上。即首先定义各局部应用的概念结构，然后将它们集成起来，得到全局概念结构。经常采用的策略是自底向上方法，即自顶向下地进行需求分析，然后再自底向上地设计概念结构。

③ 逐步扩张。首先定义最重要的核心概念结构，然后向外扩充，以滚雪球的方式逐步生成其他概念结构，直至总体概念结构。

④ 混合策略。即将自顶向下和自底向上相结合，用自顶向下策略设计一个全局概念结构的框架，以它为骨架集成由自底向上策略中设计的各局部概念结构。

【问题 2】 无论采用哪种设计方法，一般都以 E-R 模型为工具来描述概念结构。请以自底向上设计概念结构的方法为例，说明绘制 E-R 图的步骤。

【解析】 第一步，首先要根据需求分析的结果（数据流图、数据字典等）对现实世界的数据进行抽象，设计各个局部视图即分 E-R 图。 在需求分析阶段，通过对应用环境和要求进行详尽的调查分析，用多层数据流图和数据字典描述了整个系统。设计分 E-R 图的第一步就是要根据系统的具体情况，在多层的数据流图中选择一个适当层次的（经验很重要）数据流图，让这组图中每一部分对应一个局部应用，即可以这一层次的数据流图为出发点，设计分 E-R 图。

一般而言，中层的数据流图能较好地反映系统中各局部应用的子系统组成，因此人们往往以中层数据流图作为设计分 E-R 图的依据。

第二步，集成局部视图。集成局部 E-R 图又分为两步：首先合并局部视图；再修改与重构。

【例题 1-21】 阅读以下有关关系代数的叙述，回答问题 1 和问题 2。

设有关系 S、SC、C，试用关系代数表达式完成下列操作。

S（snum，sname，age，sex）例：（001，'李强'，23，'男'）

SC（snum，cnum，score）例：（003，'C1'，83）

C（cnum，cname，teacher）例：（'C1'，'数据库原理'，'王华'）

【问题 1】 检索既选修了 C1 课程，又选修了 C2 课程的学生的名单。

【解析】 (Πsname(S∞cnum='C1'（SC）))∩(Πsname(S∞cnum='C2'（SC）))

【问题 2】 检索选修了“程军”老师所授课程之一的学生的名单。

【解析】 Πsname(S∞SC∞teacher='程军'(C))

【例题 1-22】 关系型数据库的特点有哪些？请解释关系数据库中以下名词的意义：关系、属性、域、元组、候选码、主码。

【解析】 关系型数据库的特点是模型简单、数据独立性高、有较为坚实的理论基础。

关系：有应用语义的二维表，表中的每一行描述事物或事物的一部分的状态的数据，表中的每一列描述事物的某个特征。

属性：二维表中的一列就是关系模式中的一个属性。表中的每一个属性必须是基本类型。表中的每一列的所有值必须是同类型、同语义的。属性的值只能是域中的值。表中的每一列都必须有唯一的名字，列在表中的顺序是不重要的。

域：属性的取值范围。

元组：二维表中的一行称为一个元组。

候选码：关系中按应用语义能唯一标识元组的最小的属性集合。

主码：指定为关系中元组标识的候选码，称主码属性组为主属性。主码有时也被称为

主关键字或主键。

【例题 1-23】 阅读以下 E-R 图相关的叙述，回答问题 1 到问题 3。

在集成 E-R 图时，各部分 E-R 图之间的组成往往会冲突。这些冲突主要有 3 类：属性冲突、命名冲突和结构冲突。

【问题 1】 属性冲突主要有哪两种，请举例说明。

【解析】

① 属性域冲突，即属性值的类型、取值范围或取值集合不同。

② 属性取值单位冲突。

【问题 2】 命名冲突主要有哪两种，请举例说明。

【解析】

① 同名异义。

② 异名同义（一义多名）。

【问题 3】 结构冲突主要有哪两种，请举例说明。

【解析】

① 同一对象在不同应用中具有不同的抽象。例如“课程”在某一局部应用中被当作实体，而在另一局部应用中则被当作属性。

② 同一实体在不同局部视图中所包含的属性不完全相同，或者属性的排列次序不完全相同。

③ 实体之间的联系在不同局部视图中呈现不同的类型。例如实体 E1 与 E2 在局部应用 A 中是多对多联系，而在局部应用 B 中是一对多联系；又如在局部应用 X 中 E1 与 E2 发生联系，而在局部应用 Y 中 E1、E2、E3 三者之间有联系。

解决方法是根据应用的语义对实体联系的类型进行综合或调整。

【例题 1-24】 数据库的三级模式指什么？分别代表什么意义？

【解析】 数据库的三级模式结构是指数据库系统是由外模式，模式和内模式三级构成。模式也称逻辑模式，是数据库中全体数据的逻辑结构和特征的描述，是所有用户的公共数据视图。外模式也称子模式或用户模式，它是数据库用户（包括应用各方和最终用户）看见使用的局部数据的逻辑结构和特征的描述，是数据库用户的数据视图，是与某一应用有关的数据的逻辑表示。内模式也称存储模式，它是数据物理和存储结构的描述，是数据在数据库内部的表示方式。

【例题 1-25】 阅读以下有关信息系统需求分析的叙述，回答问题 1 和问题 2。

设有关系 S、SC、C，试用关系元组演算表达式完成下列操作。

S(snum，sname，age，sex) 例：(001，'李强'，23，'男')

SC(snum，cnum，score)　例：(003，'C1'，83)

C(cnum，cname，teacher) 例：('C1'，'数据库原理'，'王华')

【问题 1】 检索选修了“程军”老师所授课程之一的学生的名单。

【解析】 {t(1)|(u)(v)(w)(S(u)∧SC(v)∧C(w)∧t[1]=u[1]∧u[1]=v[2]∧v[2]=w[1]∧w[3]='程军')

【问题 2】 检索年龄大于 21 的男生的学号和姓名。

【解析】 {t(2)|(r)(S(r)∧t[1]=r[1]∧t[2]=r[2]∧r[3]>21∧r[4]='男')

【例题 1-26】 实体、属性、域、实体型、实体集分别表示什么意义？

【解析】 实体：客观存在并可相互区别的事物称为实体。属性：实体所具有的某一特性称为属性。域：属性的取值范围称为该属性的域。实体型：具有相同属性的实体必然具有共同的特征和性质。实体集：同型实体的集合称为实体集。

【例题 1-27】 阅读以下有关关系模式的叙述，回答问题 1 到问题 7。

现有如下关系模式：

教师（教师编号，姓名，电话，所在部门，借阅图书编号，图书名称，借期，还期，备注）

【问题 1】 教师编号是候选码吗？

【解析】 候选码是教师编号、借阅图书编号、借期，所以教师编号不是候选码。

【问题 2】 说明对上小题判断的理由是什么？

【解析】 判断的理由是教师编号不能唯一决定元组。

【问题 3】 写出该关系模式的主码。

【解析】 该关系模式的主码是（教师编号、借阅图书编号、借期）。

【问题 4】 该关系模式中是否存在部分函数依赖？如果存在，请写出两个。

【解析】 存在着部分函数依赖。主码与教师姓名是部分函数依赖。

【问题 5】 说明要将一个 1NF 的关系模式转化为若干个 2NF 关系，需要如何做？

【解析】 对 1NF 关系进行投影，消除原关系中非主属性对码的部分依赖，将 1NF 变为 2NF。

【问题 6】该关系模式最高满足第几范式？并说明理由。

【解析】 该关系模式最高满足 1NF，因为非主属性与码间存在部分函数依赖。

【问题 7】 重新分解该关系模式，使之满足 3NF。

【解析】 重新分解后的关系模式如下：

T1（教师编号、借阅图书编号、借期）

T2（借期、还期）

T3（教师编号、姓名、电话、所在部门）

T4（图书编号、图书名称）。

【例题 1-28】 在某中小型数据工程中，刚毕业不久的大学生小刘在设计数据库时，认为如果按照第三范式和 E-R 模型图来设计，不可避免的要为保持数据一致性，在应用程序层或 DBMS 做许多工作，而不按范式设计的数据库；虽然存在数据冗余，但是却可以在开发阶段给容易一些，而且实际项目中按照范式来设计数据库的好像很少。小刘的看法有哪

些不足之处？作为一个数据库的设计人员，如何取舍呢？

【解析】 小刘的看法不完全正确，但也完全按第三范式对性能也有影响，其实使用第三范式只是减少冗余，因为冗余容易给程序带来错误，但如果觉得冗余带来的开发、维护代价可以接受，就没必要强求第三范式。一般来说，先按第三范式设计好数据库，然后再根据性能需要降低范式。

【例题 1-29】 阅读下面的叙述，回答问题 1 到问题 4。

在某个数据库系统中，要求根据各学期所开设的课程（教学任务），由不同的院（系）教学秘书为本院（系）的各年级的学生选择必修课，由学生自主选择选修课，在课程学完之后，由教师进行成绩的录入。另外，教学任务应来自于各院（系）的教学计划。通过对基本需求进行调查分析可知：学生的基本信息、包括学号、姓名、性别、系名、学级、年龄；教学计划（课程）的描述为课程号、课程名、选修/必修、课程教师号、系名、学级；课程成绩的基本信息包括学号、课程号、成绩；教学秘书的信息包括教学秘书号、教学秘书名、系名等；教师信息包括教师号、教师名、系名、年龄。

【问题 1】 将数据库系统中各个实体画成 E-R 图。

【解析】 各实体 E-R 图分别如图 1-7、图 1-8、图 1-9、图 1-10、图 1-11 所示。

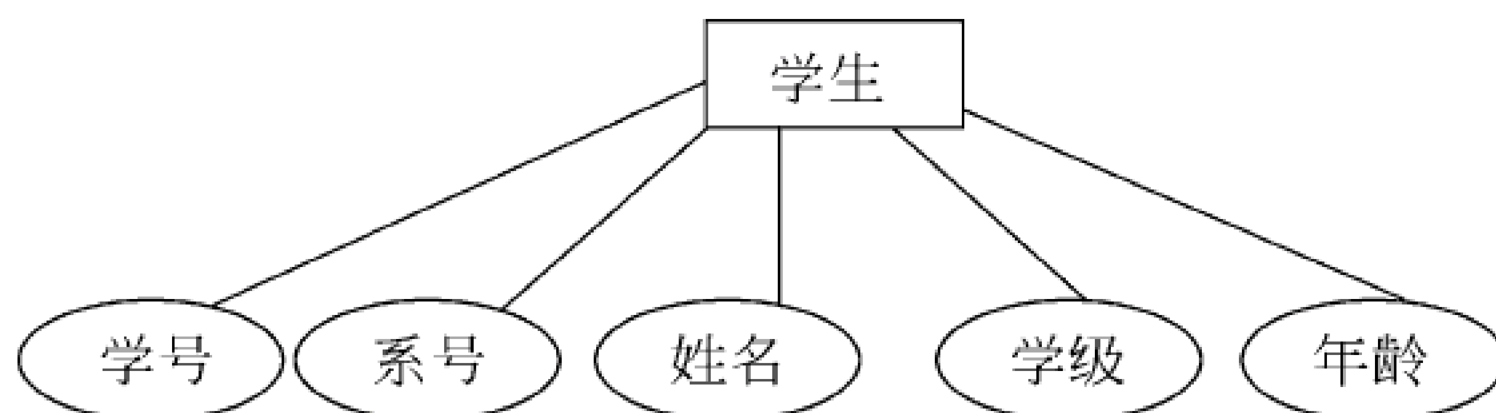


图 1-7 教学系统的 E-R 图（学生实体）

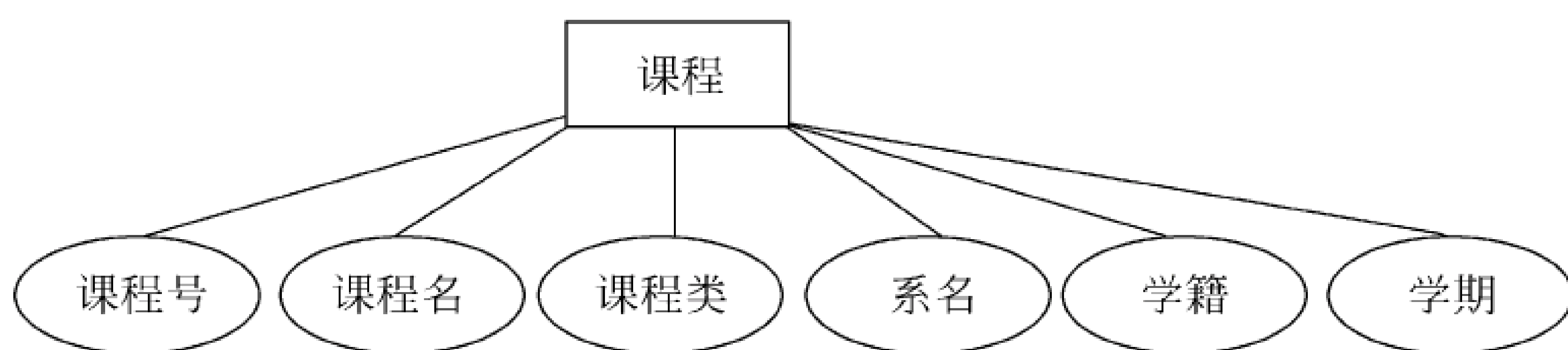


图 1-8 教学系统的 E-R 图（课程实体）

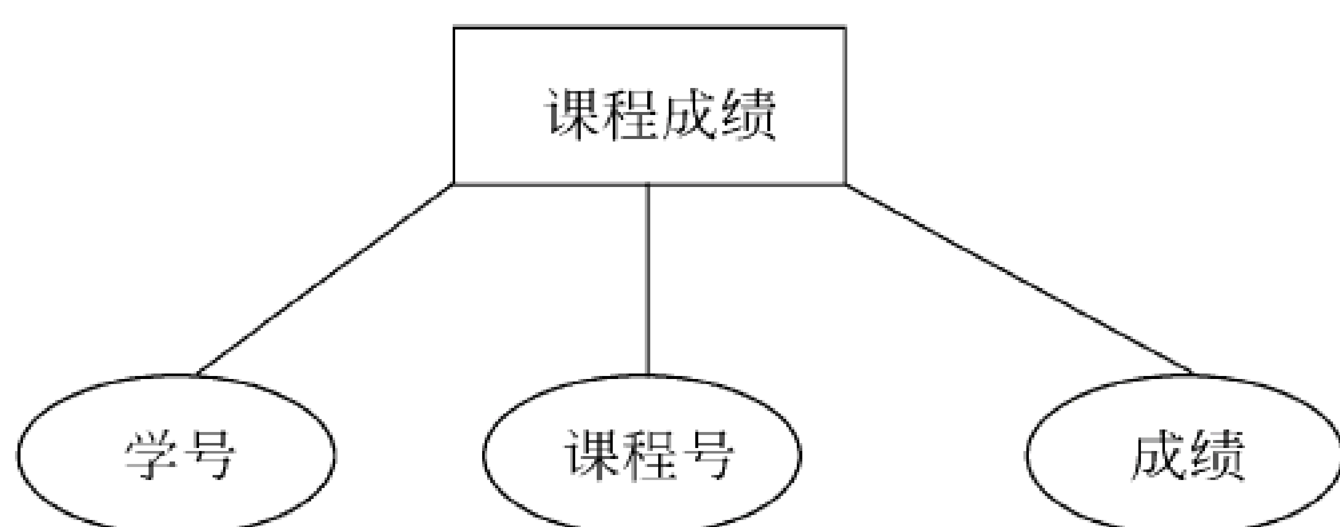


图 1-9 教学系统的 E-R 图（课程成绩实体）

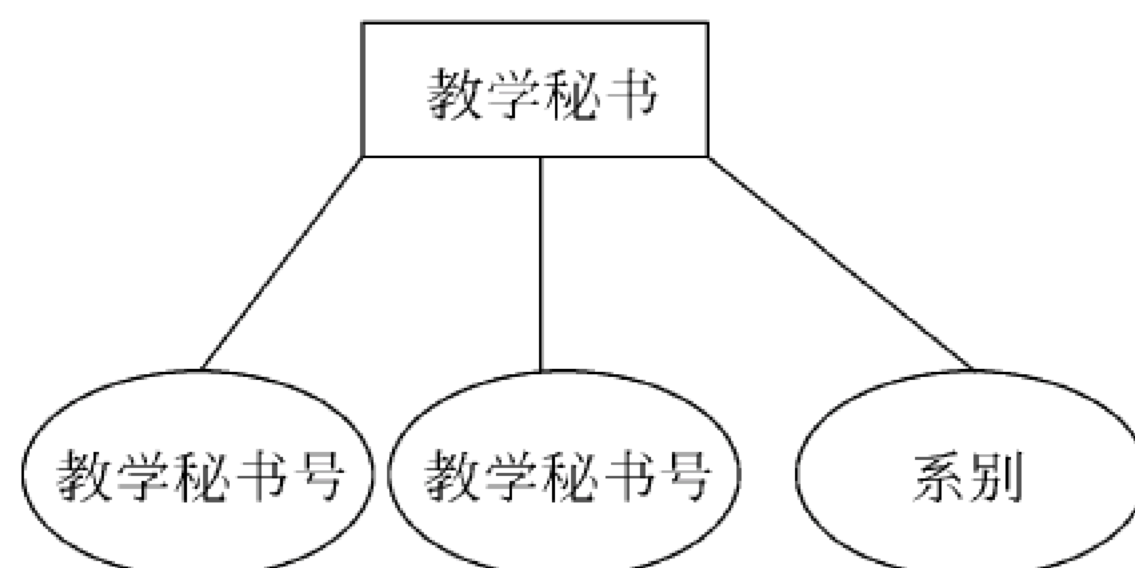


图 1-10 教学系统的 E-R 图（教学秘书实体）

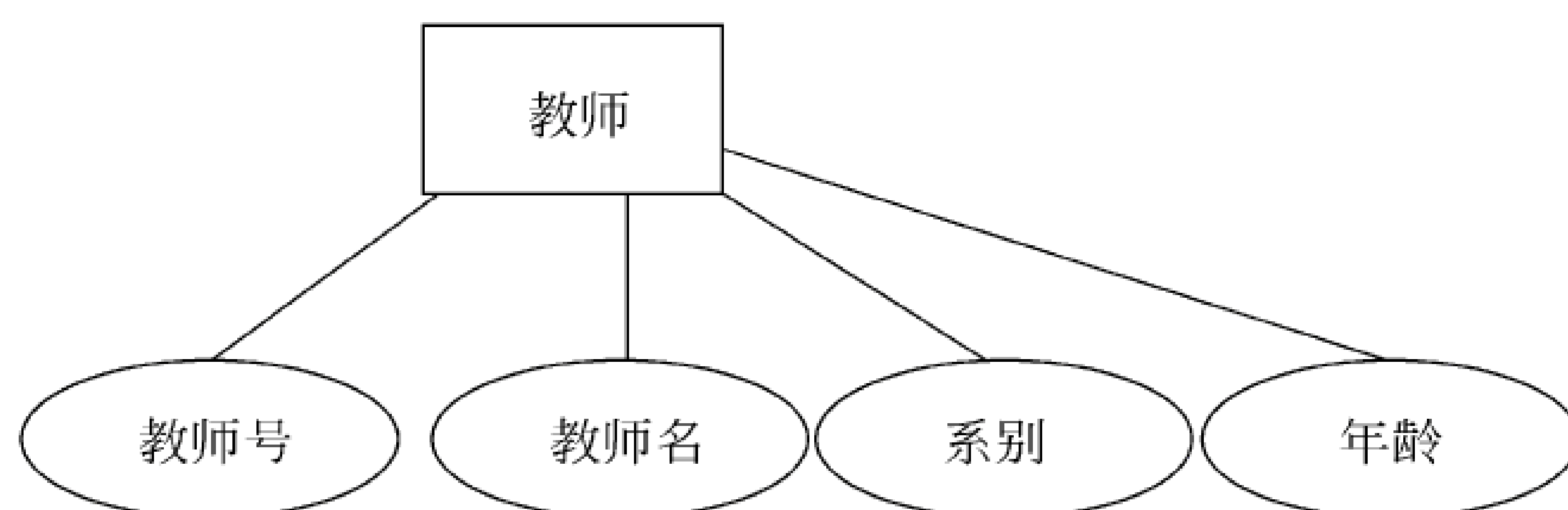


图 1-11　教学系统的 E-R 图（教师实体）

【问题 2】 根据需求将数据库系统中各个实体之间的联系画出。

【解析】 各实体间的联系图如图 1-12、图 1-13、图 1-14、图 1-15 所示。



图 1-12　教学系统的 E-R 图（学生-课程联系）



图 1-13　教学系统的 E-R 图（教学秘书-课程联系）



图 1-14　教学系统的 E-R 图（学生-课程成绩联系）



图 1-15　教学系统的 E-R 图（教师-课程成绩联系）

【问题 3】 画出合并后的 E-R 图。

【解析】 合并后的 E-R 图如图 1-16 所示。

【问题 4】 结合自己的项目经验，谈谈合并 E-R 图时的注意点有哪些？

【解析】 合并 E-R 图时不能简单地将各个分 E-R 图合到一起，而是必须消除各个分 E-R 图中的不一致，已形成一个能为全系统中所有用户共同理解和接受的统一的概念模型，

合理消除各分 E-R 图的冲突。还应注意消除不必要的冗余。

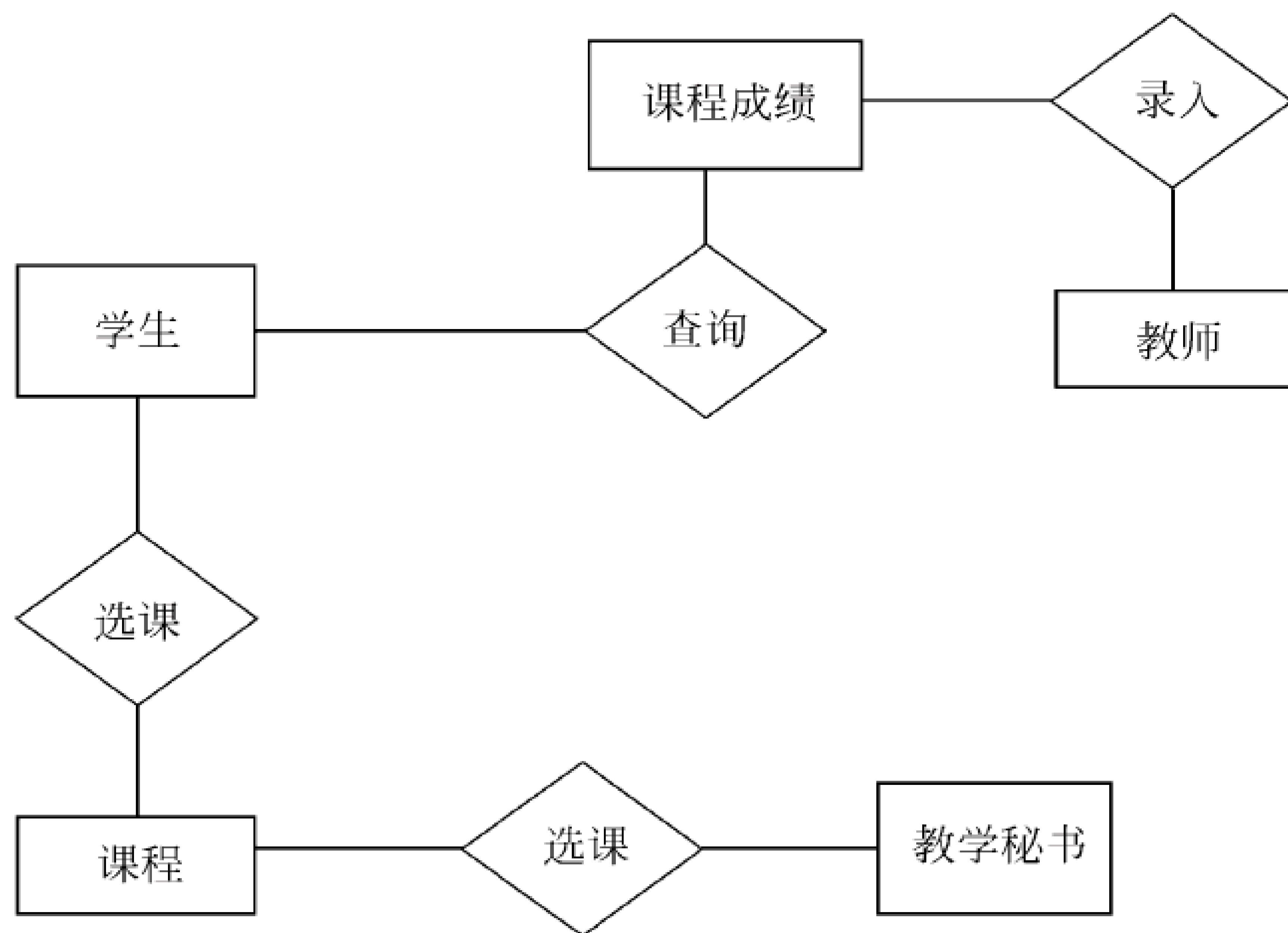


图 1-16　教学系统的 E-R 图（合并）

【例题 1-30】 阅读以下有关教学管理系统需求分析的叙述，回答问题。

一个教学管理系统中，学生可以选择多门课程且至少要选择一门，每一门课程可以由多个教师讲授，一个教室同时只能安排一门课程，但不同的时间可以安排多门课程。

学校教学管理人员和每个系的教学管理人员需要在学生选课后查看学生选课的情况，在课程结束后查看学生的成绩，但每个系的管理人员只能查看本系学生的学习情况。

【问题】画出以上系统的 E-R 图。在 E-R 图中写出每个实体的属性（只写出基本的属性），所有的联系给出两个方向的名称。（需要使用一个实体存放课程成绩，教学管理人员也需要相应的实体。）

【解析】 该系统的 E-R 图如图 1-17 所示。

【例题 1-31】 阅读以下有关教学管理系统的叙述，试完成问题 1 和问题 2 的设计。

教师：教师号、姓名、性别、职称。

课程：课程号、课程名。

工作单位：单位名、电话。

上述实体集中存在如下联系：

一个教师可以讲授多门课程，一门课程可为多个教师讲授，教师讲授的班级信息用班号表示；一个单位可以有多个教师，一个教师只能属于一个单位。

【问题 1】 构造满足需求的 E-R 图。

【解析】 满足需求的 E-R 图如图 1-18 所示。

【问题 2】 将 E-R 图转换为等价的关系模式。

【解析】 问题 1 中的 E-R 图对应的等价关系模式如下：

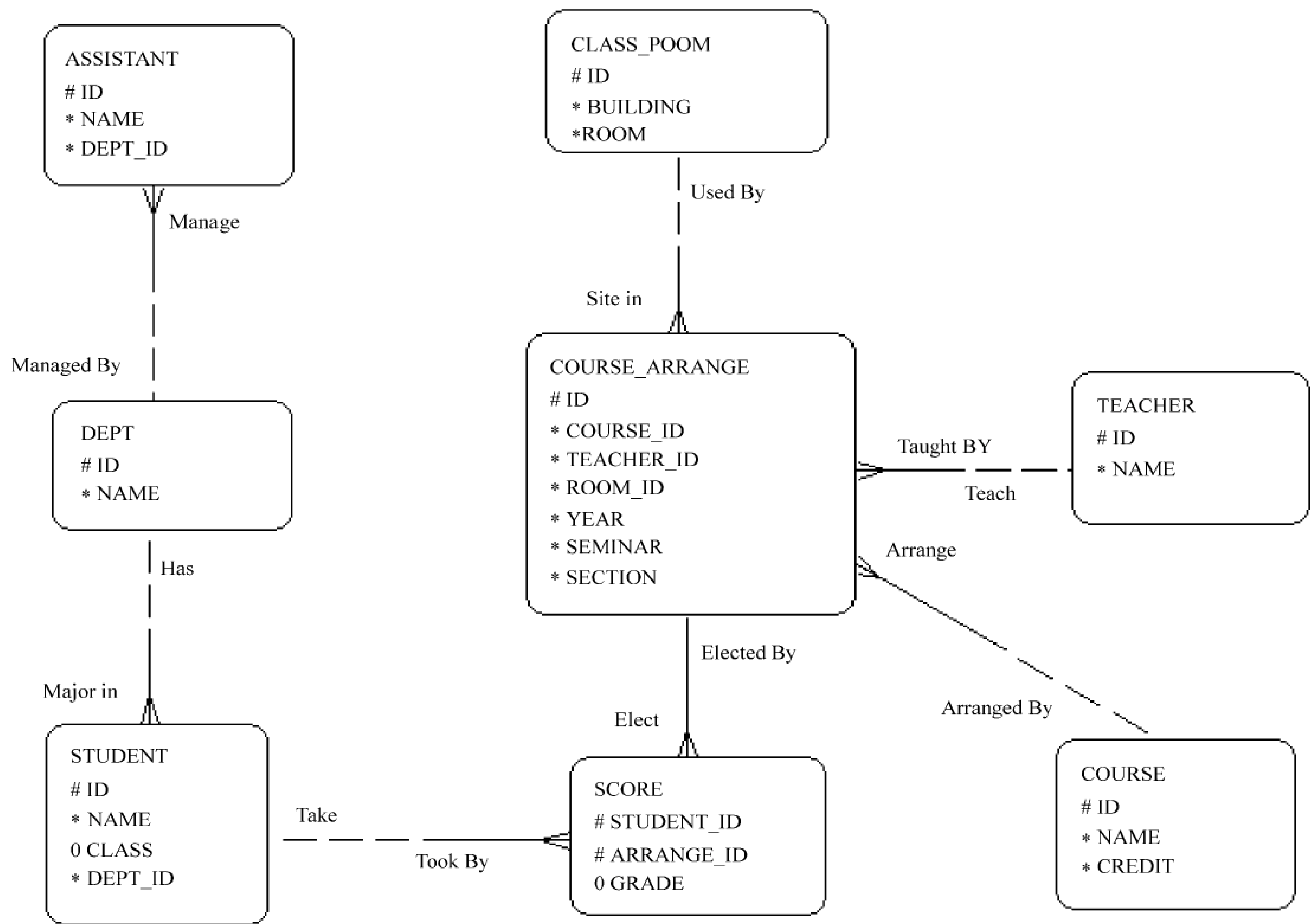


图 1-17　教学管理系统的 E-R 图

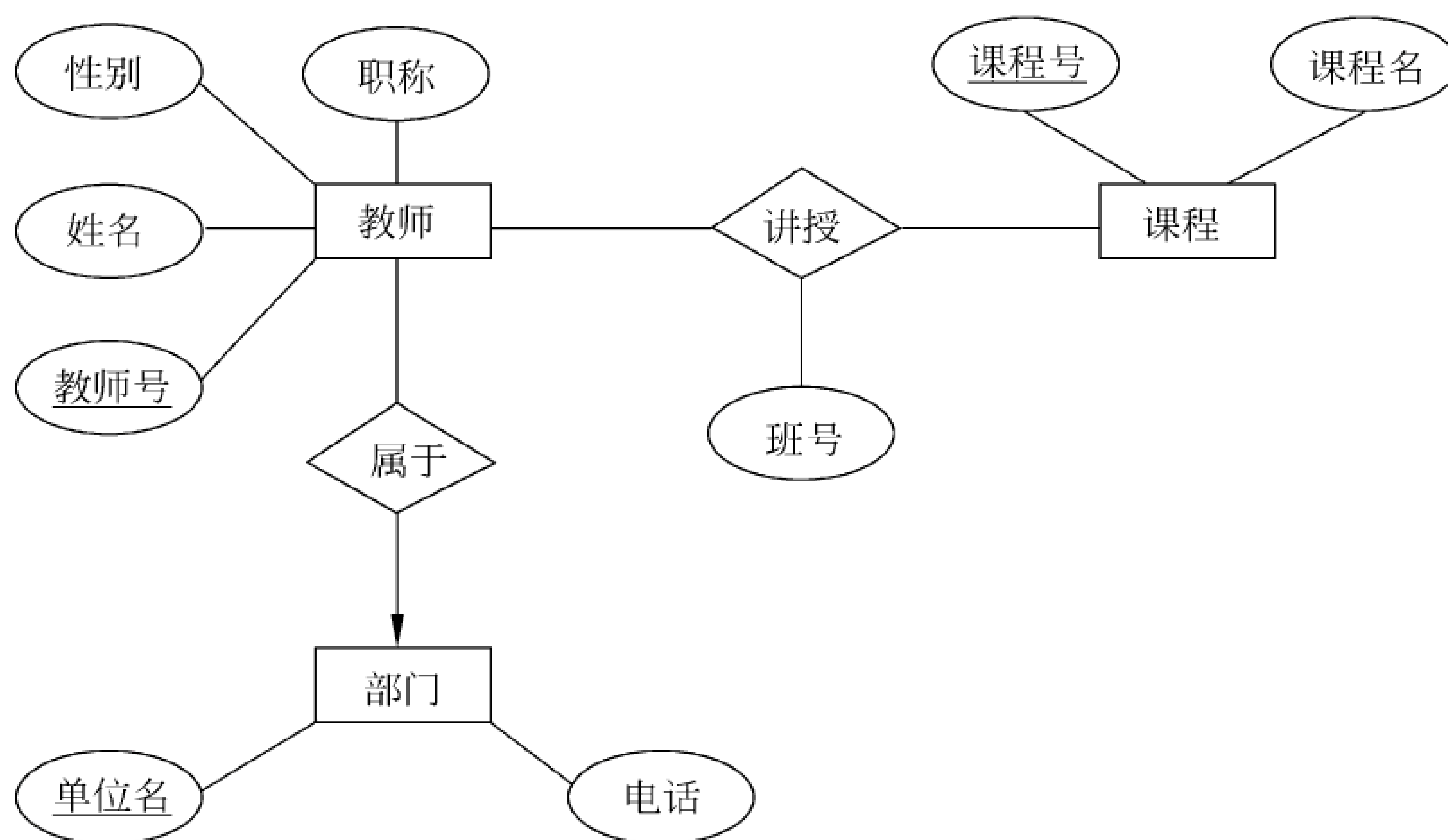


图 1-18　教学管理系统的 E-R 图

教师（教师号，姓名，性别，职称）

课程（课程号，课程名）

部门（单位名，电话）

讲授（教师号，课程号）

属于（教师号，单位名）

【例题 1-32】 阅读以下有关信息系统需求分析的叙述，回答问题 1 和问题 2。

设有关系 student(snum，sname，sdept，mname，cname，grade)，{snum，sname}为键码，设关系中有如下函数依赖：

{snum，cname}→{sname，sdept，mname}

{snum}→{sname，sdept，mname}

{snum，cname}→{grade}

{sdept}→{mname}

【问题 1】 关系 student 属于第几范式？

【解析】 关系 student 是 1NF。

【问题 2】 如果关系 student 不属于 BCNF，请将关系 student 逐步分解为 BCNF。(要求：写出达到每一级范式的分解过程，并指明消除什么类型的函数依赖。)

【解析】 首先消除部分函数依赖{snum，cname}→{sname，sdept，mname}，将关系分解为：

R1(snum，sname，sdept，mname)

R2(snum，cname，grade)

在关系 R1 中存在非主属性对键码的传递函数依赖 snum→sdept 和 sdept→mname，所以以上关系模式还不是 BCNF，进一步分解 R1：

R11(snum，sname，sdept)

R12(sdept，mname)

R11，R12 都是 3NF。

关系模式如下：

R2(snum，cname，grade)

R11(snum，sname，sdept)

R12(sdept，mname)

R2，R11，R12 关系模式存在的函数依赖如下：

snum，cname→grade　snum，sname→sname，sdept　sdept→mname

上述函数依赖都是非平凡的，并且决定因素是键码，所以上述关系模式是 BCNF。

1.4.2 物理结构设计

1．知识点提炼

（1）设计方法与内容

数据库在物理设备上的存储结构与存取方法称为数据库的物理结构，它依赖于给定的计算机系统。为一个给定的逻辑数据模型设计一个最适合应用要求的物理结构的过程，就是数据库的物理设计。

在数据库的物理结构中，数据的基本单位是记录，记录是以文件的形式存储的，一条存储记录就对应着关系模式中的一条逻辑记录。在文件中还要存储记录的结构，如各字段长度、记录长度等，增加必要的指针及存储特征的描述。

物理设计应做以下 3 方面的工作：确定数据分布、确定存储结构、确定存取方式等。

（2）存取方法的选择

选择数据的存取方法，确定数据的存储结构，即选择数据文件中记录之间的物理结构。在文件中，数据是以记录为单位存储的，可以是顺序存储、哈希存储、堆存储和 B+树存储等，要根据数据的处理要求和变更频度，选定合理的物理结构。

为提高数据的访问速度，通常会采用索引技术。在物理设计阶段，要根据数据处理和修改要求，确定数据库文件的索引字段和索引类型。

2．难点分析

从企业计算机应用环境出发，确定数据是集中管理还是分布式管理。如果是分布式管理，数据如何分布应主要考虑 3 方面的内容：一是根据不同应用分布数据，企业的不同部门一般会使用不同数据，将与部门应用相关的数据存储在相应的场地，使得不同的场地上处理不同的业务，对于应用多个场地的业务，可以通过网络进行数据处理；二是根据处理要求确定数据的分布，对于不同的处理要求，也会有不同的使用频度和响应时间，对于使用频度高、响应时间短的数据，应存储在高速设备上；三是根据数据的存储调整关系模式，对数据的分布存储必然会导致数据的逻辑结构的变化，要对关系模式作新的调整，回到数据库逻辑设计阶段作必要的修改。

3．典型例题

【例题 1-33】 简述数据库设计包含哪两方面的内容？

【解析】 结构特性设计：通常是指数据库模式或数据库结构设计，它应该具有最小冗余的、能满足不同用户数据需求的、能实现数据共享的系统。数据库结构特性是静态的，一旦形成轻易不再变动。但由于用户的需求可能会不断地更新变化，在设计时应考虑今后需求，留有扩充余地，使系统容易改变。

行为特性设计：行为特性设计是指应用程序、事物处理的设计。用户通过应用程序访问和操作数据库，用户的行为和数据库结构紧密相关。

【例题 1-34】 阅读以下有关信息系统的叙述，回答问题 1 到问题 3。

在应用系统中，尤其在联机事务处理系统中，对数据查询及处理速度已成为衡量应用系统成败的标准。在某税务部门的联机事务处理系统应用中，系统的某功能查询速度太慢，用户体验为“死机”。经该单位的信息与自动化技术科初步断定，问题的在于数据库的物理设计不合理。

【问题 1】 一般来说，数据库的物理设计通常应该考虑哪些问题？结合自己的项目经验，谈谈解决这些问题的关键点。

【解析】 解决这些问题的关键点如下所述。

① 确定数据的存储结构。确定数据库存储结构时要综合考虑存取时间、存储空间利用率和维护代价 3 方面的因素。这 3 个方面常常是相互矛盾的，例如消除一切冗余数据虽然能够节约存储空间，但往往会导致检索代价的增加，因此必须进行权衡，选择一个折中方案。

② 设计数据的存取路径。在关系数据库中，选择存取路径主要是指确定如何建立索引。例如，应把哪些域作为次码建立次索引，建立单码索引还是组合索引，建立多少个为合适，是否建立聚簇索引等。

③ 确定数据的存放位置。为了提高系统性能，数据应该根据应用情况将易变部分与稳定部分、经常存取部分和存取频率较低部分分开存放。

④ 确定系统配置。DBMS 产品一般都提供了一些存储分配参数，供设计人员和 DBA 对数据库进行物理优化。初始情况下，系统都为这些变量赋予了合理的默认值。但是这些值不一定适合每一种应用环境，在进行物理设计时，需要重新对这些变量赋值以改善系统的性能。

【问题 2】 经过该单位的信息与自动化技术科进一步分析发现，系统查询缓慢的主要原因是数据的存取路径设计时未考虑周全，一些重要的数据表未建立聚簇索引或建立了聚簇索引但选择的列不正确。结合自己的项目经验，选择聚簇索引的候选列应注重考虑哪些列？

【解析】

① 主键列。

② 按范围存取的列。

③ 在 GROUP BY 或 ORDER BY 中使用的列。

④ 不经常修改的列。

⑤ 在连接操作中使用的列。

【问题 3】 该单位的信息与自动化技术科的王科长还建议将表和索引分别放在不同的磁盘上，以提高数据库系统的查询效率，你认为该措施的原理是什么？

【解析】 在查询时，由于两个磁盘驱动器分别在工作，因而可以保证物理读写速度比较快。

【例题 1-35】 阅读以下有关数据库文件之间的 I/O 竞争的叙述，回答问题。

数据库文件之间的 I/O 竞争是数据库之大忌，所以对数据库规划之前要先对数据文件的 I/O 进行初步的评估。通常情况下，应用的产品数据库表所在的表空间会很活跃，索引表空间和数据字典之类的表空间也很活跃的，对于事物比较频繁的应用中，重做表空间也很经常，所以对不同类型的数据库其数据文件的 I/O 竞争也会略有不同。

【问题】 根据自己的项目经验，为避免数据库文件之间的 I/O 竞争，谈谈数据库物理设计基本原则有哪些？

【解析】

① 应用的表和索引通常应该被分配或分区到多个表空间中，以降低单个数据文件的 I/O，最好把每一种功能相同的区域对象建立单独的表空间。

② 不要把除数据字典表和系统回退段外的其他东西放到系统表空间中，要把能移出系统表空间的对象都移出。

③ 索引段不应该和相关表放在同一表空间中，因为它们在数据管理和查询时会产生很多的并发 I/O。

④ 临时表空间是用以存储大量的排序，所以其他的应用对象是不能放在临时表空间的。

【例题 1-36】 阅读以下关于企业成本核算管理软件方面的叙述，回答问题。

在某通信产品制造工厂的财务管理系统设计中，十分重视成本核算与管理模型的分析，根据该厂通信产品制造流程的具体特点，采用相应的各类成本计算方式，需要在每一个流程环节上计算出产品制造过程中所涉及到的成本；同时要依照产品的产量、质量、各类部件成本和各类耗费等计划数据，具体编制出产品生产成本计划，从而进行成本的分析与考核，图 1-19 是成本核算管理软件的基本功能模块结构简图。

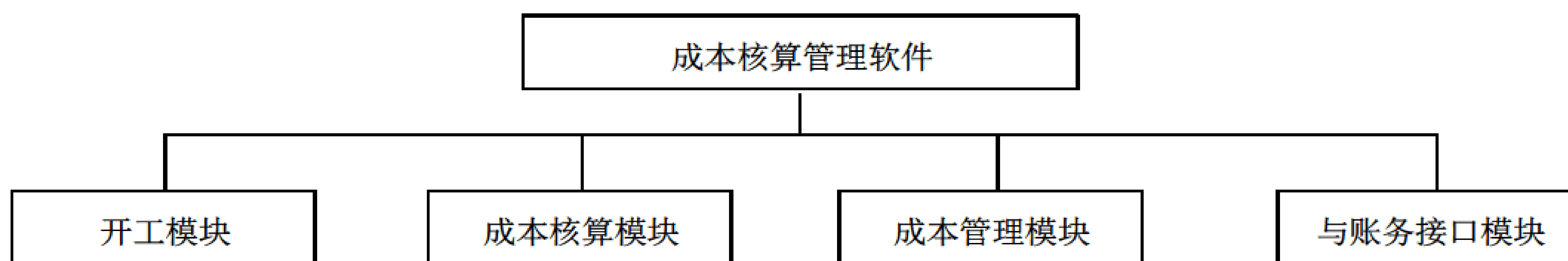


图 1-19　成本核算管理软件的基本功能

① 开工模块——用来提供各类通信产品的产量、入库数量、各类零部件与材料的计划单价、各个制造环节中每一部分的单耗与工费等。

② 成本核算模块——又可按生产工序、车间与全厂细分为几个成本核算子模块。其主要任务包括计算出各类产品的总成本和单位成本，登记并打印各类账表，以及统计出有关的成本台账等。

比如，经过各类处理后，可能产生出：各类产品的车间成本与厂部成本；各级有关费用与管理费明细表；可比产品成本表与不可比产品成本表；生产费用表；成本项目汇总表；各类成本台账等。

③ 成本管理模块——其主要任务可区分为 3 个子模块：成本计划、成本分析与成本考核。例如，通过调用开工模块与成本核算模块中的有关数据，经处理后可以获得：各级各类成本计划；各级成本分析与成本项目分析；费用增减因素分析；各级责任成本考核情

况表等。

④ 接口模块——负责成本模型与账务模型之间的相互连接。比如可以把成本核算的处理结果形成转账凭证，并被自动转入账务系统处理。通常可以用“成本核算底稿”为中心来考虑接口与连接。

【问题】 在该厂成本核算管理软件中涉及到的数据量较大，数据的重复使用频度较高。经分析，在成本核算管理软件中涉及的主要数据可分为以下两大类。

① 原始型数据，包括从账务系统中转来的数据和本软件需要输入的数据（比如各环节生产费用、当月产量等）。

② 对原始型数据进行处理过程中生成的数据（比如累计产量、单位成本、总成本等）。

为了节省计算机存储空间，加快处理速度，提高数据共事能力，又能方便操作，请以 100 字以内文字，提出两条节省联机存储数据量的措施。

【解析】 一提高各个模块的数据共享力度；二只存储原始型数据和对原始型数据进行处理过程中生成的数据中常用的数据（例如常被查询的数据）。

【例题 1-37】 阅读以下关于数据库应用系统性能优化方面的叙述，回答问题 1 和问题 2。

某市经济信息中心采用 Oracle 数据库建立起了一个经济信息统计方面的大型数据库应用系统，尽管配置了相当优良的硬件和网络环境，但实施后的整体性能表现较差。特别是随着业务量与信息量的迅速扩大，数据库系统的存取速度显著减慢，存储效率也明显下降。该信息中心通过反复实践与摸索，并邀请数据库专家一起会诊，认为可能从以下 4 个方面进一步优化数据库应用系统。

① 通过调整服务器配置、操作系统配置与数据库管理系统的有关参数，优化系统的 I/O 性能（因为数据库应用中最主要的查询与修改数据操作大多须通过 I/O 来完成），尤其是改进磁盘 I/O 的效率与性能。

② 优化“索引”的建立与使用机制，尽可能提高数据查询的速度或效率。

③ 合理使用聚类（Cluster），改进查询响应时间和系统的综合性能。这里的“聚类”是把单独组织的、但在逻辑上经常需要连接的、较为稳定的几个基本表聚集在一起（在物理上实现邻近存放），可以显著减少数据的搜索时间，从而提高性能。

④ 优化服 L 查询，比如：优化相应的表连接，优化嵌套子查询，避免全表的反复查询，避免嵌套的游标（Cursor）和多重循环等。

【问题 1】 对于大型数据库系统来说，索引机制可以在很大程度上提高数据查询的效率。但是必须注意的是索引是以空间为代价来换取时间的，而且一般仅当表的容量较大时，才能显现出索引的作用。

以下是该信息中心的技术人员在讨论优化索引的使用时提出过的两条主要建议。

① 尽可能充分地去理解索引的基本原理和在本系统中使用索引时的一些规则，从而为正确使用索引奠定基础。比如：在某些调词后不使用索引（如 ORNOTIN 等），查询的记录数超过表总记录数的 20%以上时不适宜使用索引，在另一些谓词后出现的索引字段会使

用索引等。

② 实践已证明，在有很多基本表的场合下，由多名开发人员大量创建不尽合理的索引，可能会显著降低系统的性能。根据应用的实际需要，合理地创建“索引”，在本系统应用中有可能考虑一些创建索引的原则或指导性规则。

- 原则上，对记录数较多的表，应尽可能考虑创建索引的可行性；
- 在连接多个表的场合下，最好考虑采用索引；
- 无相同值的字段可建立起“唯一性索引”（这时查询代化性能特别出色）；
- 对于只读的表可建立较多的索引，对于更新频度较高的表只能作较少索引；
- 尽可能在数据加载之后再建立索引，以避免重新索引的开销；
- 建立和执行复合索引时，应把最常用的索引序段放在前面，即保持相对合理的索引序段次序。

此外，在讨论中，一致认为选取合适的“索引字段”，对于索引实现的效率具有相当重要的作用。

请用 100 字以内的文字，指出选用具有哪些特征的字段作为索引手段最为合适？（提供一些原则）

【解析】 一般来说，具有如下特征的字段应作为索引的候选字段：主键字段，按范围存取的字段，在 GROUP BY 或 ORDER BY 中使用的字段，不经常修改的字段，在连接操作中使用的字段。

【问题 2】 上面已经指出，系统 I/O 经常会成为制约数据库应用系统性能的瓶颈；减少 I/O 资源的争用和合理分布各类存储空间，通常能有效地提高全系统的运行效率。比如可以使应用系统的空间与系统表空间分离，在不同的盘驱动器上设立数据表空间与索引表空间，自动整理与减少空间碎片等等。

许多开发技术人员都进一步谈到由于缺乏经验，一般在建立实体（如基本表、索引、表空间等）时，都习惯于使用 DBMS 提供的“默认参数”，结果有时会引起应用系统中性能很差的隐患。

请用 150 字以内文字简要说明为什么有时必须重新设置这些 DBMS 的默认参数？

【解析】 DBMS 产品一般都提供了一些存储分配参数，供设计人员和 DBA 对数据库进行物理优化。初始情况下，系统都为这些变量赋予了合理的默认值。但是这些值不一定适合每一种应用环境，在进行物理设计时，需要重新对这些变量赋值以改善系统的性能。通常情况下，这些配置变量包括：同时使用数据库的用户数，同时打开的数据库对象数，使用的缓冲区长度、个数，时间片大小、数据库的大小，装填因子，锁的数目等等，这些参数值影响存取时间和存储空间的分配，在物理设计时就要根据应用环境确定这些参数值，以使系统性能最优。 在物理设计时对系统配置变量的调整只是初步的，在系统运行时还要根据系统实际运行情况做进一步的调整，以期切实改进系统性能。

1.4.3　数据库实施与维护

1．知识点提炼

（1）数据加载与应用程序调试

完成数据库的设计和应用程序的设计之后，开发人员根据设计，用选定的 RDBMS 提供的 SQL 语言的数据定义语言（DDL）及其他高级语言对设计进行代码编写，经过调试运行后，就建立了系统数据库的结构，包括数据库及基本表、索引、约束等数据库对象也都建立起来了。接下来就要加载实验数据，进行数据库系统的试运行。经过试运行后，如果系统的各项功能都已经实现，并且系统性能达到预定的要求，就可以卸载实验数据，加载用户数据，使系统正式运行。如果用户数据是旧系统的数据，不一定能够完全满足新系统的数据要求，需要对其进行处理，同时还要做好新系统的数据库的转储和恢复工作，以免发生故障时丢失数据。

（2）数据库试运行

数据库系统实现后，加载数据进行试运行。试运行阶段的测试工作一般由数据库设计人员、应用开发人员和用户联合进行，又称为联合测试。系统试运行阶段需要通过运行应用程序，执行对数据库的各种操作，测试应用程序的各项功能，测量系统的各项性能指标，分析是否实现预定的设计目标。试运行阶段应记录系统出现的各种问题，采取回溯的方式改进和完善设计和应用程序，以解决测试出现的各种问题。

（3）数据库运行与维护

数据库运行时，需要对数据和业务持续性进行监控。

动态地掌握数据库的运行状态，记录并分析系统运行信息，称为监控数据，它是发现系统问题和改进系统性能的依据。按照监控的类型，监控数据可分为性能监控数据、故障监控数据和安全监控数据。性能监控数据包括磁盘使用信息（碎片量、剩余空间、日志文件增涨情况），I/O 操作数量、频度及响应时间，缓冲区命中率，事务量及锁状况。通过分析这些数据，找出影响性能的问题所在，为下一步性能调整提供依据。分析故障监控数据，可以找出故障的原因，是事务处理程序的内部错误，还是系统调度的问题，以及是否因为系统硬件故障，做出相应的处理。安全监控数据主要是记录用户对数据库的访问和修改操作，可以通过日志文件来得到，判定是否有未授权用户的存取，分析安全漏洞的原因，对用户管理和应用程序加以改进。

业务持续性是指一个组织的主要业务流程、营运服务，以及 IT 服务能够得到连续性处理。在一个突发事件中，公司的主要业务、服务流程、设备、人员等因素都有着各自的持续性要求。公司的 IT 部门和其他职能部门必须相互配合工作，不仅仅体现在业务持续的计划中，更需要在具体的实施过程中得到实现。

数据库系统运行过程中，会因为某些原因而对数据库的结构做出修改，称为数据库重构。重构包括修改表结构和视图。表结构的修改有数据列的增删和修改、约束的修改、表

的分解与合并。在数据库重构过程中引入或修改视图，可能会影响数据的安全性，必须对视图进行评价和验证，保证不能因为数据库的重构而引起数据的泄密。在数据库重构的过程中，对数据库所作的修改，必须在文档中体现出来，文档必须与系统保持高度的一致性。

2．难点分析

应用程序的调试是每一位开发人员必须掌握的一项技能。调试的任务是根据系统测试时所发现的错误，找到出错原因和错误的具体位置，进行改正。系统测试工作应该避免由原开发软件的人员或小组承担，测试工作应由专门人员来进行，，而调试工作主要由程序开发人员承担。

常用的调试方法有下面几种：

① 试探法。调试人员分析错误的症状，猜测问题的所在位置，利用在程序中设置输出语句，分析寄存器、存储器的内容等手段来查找错误位置，一步步地试探和分析错误所在。该方法的缺点是效率很低，适合于结构比较简单的程序。

② 回溯法。从发现错误症状的位置开始，沿着程序的控制流程往回跟踪代码，直到找出错误根源为止。该方法适合于小型程序。

③ 归纳法。归纳法就是从测试所暴露的问题出发，收集所有正确或不正确的数据，分析它们之间的关系，提出假象的错误原因，用这些数据来证明或反驳，从而查出错误所在。

④ 演绎法。根据测试结果，列出所有可能的错误原因，分析已有的数据，排除不可能和彼此矛盾的原因，对余下的原因，选择可能性最大的，利用已有的数据完善该假设，使假设更具体。用假设来解释所有的原始测试结果，如果能解释这一切，则假设得以证实，也就找出错误；否则，要么是假设不完备或不成立，要么有多个错误同时存在，需要重新分析，提出新的假设，直到发现错误为止。调试程序时，要综合运用以上几种调试方法，以便迅速定位出错位置。

3．典型例题

【例题 1-38】 阅读以下有关数据库数据结构定义的叙述，回答问题 1 和问题 2。

确定了数据库的逻辑结构与物理结构后，就可以用所选用的 DBMS 提供的数据定义语言（DDL）来严格描述数据库结构。数据定义语言（DDL）是用来定义和删除数据库以及数据库实体的，这些命令主要是被数据库管理者用来建立和删除数据库实体的。

【问题 1】 请列出 4 种基本 DDL 命令。

【解析】

① CREATE 建立数据库实体。

② USE 在数据库管理系统中指定要操作的数据库。

③ ALTER 改变数据库的结构，例如修改某个表的结构。

④ DROP 从数据库管理系统中移去整个数据库实体。

【问题 2】在当前的数据库中建一个名为 student_info 的表。表包含 3 个非空属性：

first_name 和 last_name 为长度为 20 的字符，student _id 为整型。请写出 DDL。

【解析】 CREATE TABLE student _info (first_name CHAR(20) NOT NULL，last_name CHAR(20) NOT NULL，student_id INT NOT NULL)

【例题 1-39】 数据库结构建立好后，就可以向数据库中装载数据了。组织数据入库是数据库实施阶段最主要的工作。对于数据量不是很大的小型系统，可以用人工式方法完成数据的入库，一般来说，数据人工装载的步骤分为哪几步，每一步的具体工作是什么？

【解析】

① 筛选数据。需要装入数据库中的数据通常都分散在各个部门的数据文件或原始凭证中，所以首先必须把需要入库的数据筛选出来。

② 转换数据格式。筛选出来的需要入库的数据，其格式往往不符合数据库要求，还需要进行转换。这种转换有时可能很复杂。

③ 输入数据。将转换好的数据输入计算机中。

④ 校验数据。检查输入的数据是否有误。

1.4.4 数据库的保护

1. 知识点提炼

（1）数据库的备份与恢复

在数据库的运行过程中，难免会出现计算机系统的软、硬件故障，这些故障会影响数据库中数据的正确性和完整性，甚至破坏整个数据库系统，使数据库中全部或部分数据丢失。为了使数据库具有可恢复性，基本原则对数据库进行备份。系统出现故障之后，利用数据库的备份文件，及时使数据库恢复到故障前的正确状态，就是数据库恢复技术。

数据库系统可能发生的故障大致可以分为以下几类。

① 事务故障。事务故障是由于事务程序内部错误而引起的，这些错误有些是可以预期的，有些则是不可预期的，事务故障通常指非预期的故障。对于可预期的错误，可以由应用程序以回滚的方式来恢复，非预期的故障应用程序无法处理，可以由 DBMS 系统来实现故障恢复的。

② 系统故障。系统故障是指造成系统停止运行，使得系统需要重新启动的任何事件。系统故障只是丢失了数据缓冲区中的内容，影响正在执行的所有事务，但不会破坏数据库。由于系统故障中止了事务的执行过程，破坏了事务的原子性，缓冲区中的内容可能部分已写入数据库，系统重启后数据库可能处于不一致状态。

③ 介质故障。介质故障是指数据库的存储介质发生故障，这种故障直接破坏数据库，会影响到所有正在读取这部分数据的事务。

在数据库发生故障后要能够将数据库恢复到正确状态，必须建立冗余数据，在故障发生后利用这些冗余数据恢复数据库。建立冗余数据常用的技术是数据转储和建立日志文件。数据转储，又称为数据备份，即将数据库复制到另一个磁盘或磁带上。数据转储又分为静

态转储和动态转储。日志文件是用来记录事务对数据库系统的更新操作的文件。在数据库运行过程中，系统把事务开始、事务结束以及对数据库的插入、删除和修改的每一次操作作为一条记录写入日志文件中。每条记录都包括执行操作的事务标识、操作类型、更新前后的数据值、更新日期和更新时间。

有了数据转储和日志文件，就可以在系统发生故障后进行恢复。故障恢复包括撤销事务（UNDO）和重做事务（REDO）两个操作。撤销事务是指将未完成的事务撤销，使数据库回复到事务执行前的正确状态。撤销事务的具体过程如下：反向未完成的事务日志（由后向前扫描），查找事务的更新操作；对该事务的更新操作执行逆操作，用日志文件记录中更新前的值写入数据库，插入的记录从数据库中删除，删除的记录重新插入数据库中；继续反向扫描日志文件，查找该事务的其他更新操作并执行操作直至事务开始标志。重做事务是指将已经提交的事务重新执行。重做事务的过程如下：从事务的开始标识起，正向扫描日志文件，重新执行日志文件登记的该事务对数据库的所以操作，直至事务结束标识。

对于不同的故障，应该采取不同的恢复策略，具体如下：事务故障的恢复是通过撤销（UNDO）产生故障的事务，使数据库恢复到该事务执行前的正确状态来完成的，由系统自动完成，对用户是透明的。系统故障的恢复就是撤销故障发生时未完成的事务，重做（REDO）已提交的事务，系统故障的恢复是在系统重启之后自动执行的。介质故障的恢复需要重装数据库，装载故障前最近一次的备份和故障前的日志文件副本，再按照系统故障的恢复过程执行撤销和重做来恢复，介质故障的恢复需要系统管理员（DBA）的参与，系统管理员装入数据库的副本和日记文件的副本，再由系统执行撤销和重做操作。

概括一下，数据库的备份和恢复的方法如下：周期性地转储整个数据库，常称为“倒库”，即将数据库保存到磁带或活动磁盘等可脱机的介质上；建立日志文件，记录下每次数据更新（增、删、改）前后的数据值和执行更新操作的程序标识；一旦发生故障，根据数据库备份和日志文件把数据库恢复到最近的正确状态。

（2）数据库的安全性

数据库的安全性是指保护数据库以防止不合法的使用所造成的数据泄露、更改或破坏。安全性分为系统安全和环境安全两大类。为实现系统安全、防止非授权用户存取造成数据泄密或人为破坏，可以采取用户标识和鉴定、分级授权、数据加密等措施。环境安全是指如何有效地防止天灾人祸等意外事故而采取的防范措施。

对数据库的不合法的使用称为数据库的滥用，可分为无意的和恶意的。无意的滥用在事务处理时容易发生系统故障、出现异常现象以及违反数据完成性约束等逻辑错误。恶意的滥用主要是指未经授权的读取（偷窃信息）和未经授权的修改操作（破坏数据）。数据库的完整性是指尽可能避免对数据库的无意滥用；数据库的安全性是指尽可能地避免对数据库的恶意滥用。完全避免恶意滥用是不可能的，但是应该尽量增加一些保护措施，提高数据库的安全性。

安全性措施可以从物理环境、网络环境、操作系统、数据库系统和人员管理方面考虑，

数据库系统的安全性措施是我们的重点。数据库系统的安全措施主要包括权限机制、视图机制和数据加密等。权限机制即限定用户对数据的操作权限，把数据的操作限定在具有指定权限的用户范围内。视图机制是指用户或应用程序只能通过视图来操作数据，从而可以保证视图之外的数据的安全性。数据加密是指对数据库中的数据进行加密，以防数据在存储和传输过程中失密。

审计功能把用户对数据库的所有操作自动记录下来放入审计日志中。DBA 可以利用审计跟踪的信息，重现导致数据库现有状况的一系列事件，找出非法存取数据的人、时间和内容等。

按照可信计算机系统评估标准关于可信数据库系统的解释（Trusted Databas Interpretation/Trusted Computer System Evaluation Criteria，TDI/TCSEC）标准中安全策略的要求，审计功能是 DBMS 达到 C2 以上安全级别必不可少的一项指标。

（3）数据库的完整性

数据库的完整性是对数据的正确性和一致性的测度，完整性就是正确性、准确性和有效性。数据库的完整性可分为两类：域完整性控制和关联完整性控制。域是某个特定字段中允许的数据元素的类型和范围，域完整性控制就是对某个或者某几个特定字段中允许的数据元素的类型和范围加以控制。关联完整性规定的约束条件用于防止对数据进行增、删、改操作时，破坏本身及其他数据的完整性。DBMS 提供了完整性约束机制，通过对数据库表结构进行约束，当对数据进行修改时由系统对修改数据进行完整性检查，防止错误数据进入数据库。完整性约束条件作用的对象可以是表、行和列。列级约束主要是指对列的类型、取值范围、精度、非空值、值不可重复等的约束条件；行级约束是记录字段值之间联系的约束条件；表级约束是表的主码约束、表与表间的参照完整性约束、表中记录间的联系约束。

2．难点分析

数据库的并发控制。

数据库是一个共享资源，供多个用户同时使用，需要并行执行多个事务。在事务并行处理的过程中，因为多个事务对相同数据的访问，各个事务之间相互干扰，可能会导致数据的不一致性。为了避免出现错误的数据甚至破坏数据库的一致性，因此必须对多用户的并发操作加以控制、协调以保证数据的正确。常用的保护方法包括以独占方式打开数据库和对数据库文件或记录加锁，这里主要讨论并发操作。并发操作引起的数据不一致性有 3 类：丢失修改、不可重复读和读脏数据。并发控制的方法有加锁（Lock）和封锁协议两种。

加锁，即在事务执行时限制其他事务对数据的读取。并发处理多个事务时，如果对数据的读写不加以控制，很容易破坏事务的隔离性和数据的一致性。加锁就是当一个事务在对某个数据对象（可以是数据项、记录、数据集、以至整个数据库）进行操作之前，必须获得相应的锁，以保证数据操作的正确性和一致性。

在并发控制中引入两种锁：排它锁（eXclusive Locks）简称 X 锁和共享锁（Share Locks）简称 S 锁。排它锁又称写锁，用于对数据进行写操作时进行锁定，其采用的原理是禁止并发操作。当事务 T 对某个数据对象 R 实现 X 封锁后，其他事务要等 T 解除 X 封锁以后，才能对 R 进行封锁。这就保证了其他事务在 T 释放 R 上的锁之前，不能再对 R 进行操作。

共享锁又称为读锁，用于对数据进行读操作时进行锁定，其采用的原理是允许其他用户对同一数据对象进行查询，但不能对该数据对象进行修改。当事务 T 对某个数据对象 R 实现 S 封锁后，其他事务只能对 R 加 S 锁，而不能加 X 锁，直到 T 释放 R 上的 S 锁。这就保证了其他事务在 T 释放 R 上的 S 锁之前，只能读取 R，而不能再对 R 作任何修改。

通过对数据加锁，虽然可以限制其他事务对数据的访问，但是降低了事务的并发性。同时还要避免并发访问的几个应用程序无休止地等待其他程序释放自己所需的数据单元而造成的死锁现象。

实际上，锁是一个控制块，其中包括被加锁记录的标识符及持有锁的事务的标识符等。在加锁时，要考虑一定的封锁规则，例如，何时开始封锁、封锁多长时间、何时释放等，这些封锁规则称为封锁协议。封锁协议是对数据加锁类型、加锁时间和释放锁时间的一些规则的描述。通过封锁协议可以在保证事务的一致性的前提下尽可能地提高事务的并发性。封锁协议分为以下 4 种。

① 一级封锁协议，即事务 T 在修改数据 A 之前必须先对其加 X 锁，直到事务结束才释放 X 锁。一级封锁协议使得在一个事务修改数据期间，其他事务不能对该数据进行修改，只能等到该事务结束后，解决了丢失修改的问题。

② 二级封锁协议，即在一级封锁协议加上事务 T 在读取数据 A 之前必须对其加上 S 锁，读完后即可释放 S 锁。二级封锁协议使得一个事务不能读取被其他事务修改中的数据，解决了读脏数据的问题。但是，如果事务 T 在读取数据 A 之后，其他事务再对 A 作完修改，事务 T 再读取 A，还会产生不可重复读的错误的。

③ 三级封锁协议，即在一级封锁协议加上事务 T 在读取数据 A 之前必须对其加上 S 锁，直到事务结束才释放 S 锁。三级封锁协议使得一个事务读取数据期间，其他事务只能读取该数据而不能修改，解决了不可重复读的问题。

④ 两段锁协议，即对任何数据进行读写之前必须对该数据加锁；在释放一个封锁之后，事务不再申请和获得任何其他封锁。两段锁协议缩短了持锁时间，提高了并发性，同时又解决了数据的不一致性问题。

数据库的并发控制是本小节的重点和难点，考生应给予足够的重视。

3．典型例题

【例题 1-40】 阅读以下关于数据库事务方面的叙述，回答问题。

事务是定义和维护一致性的单位，封锁就是要保证这种一致性。如果对封锁的要求高会增加开销，降低并发性和效率；有的事务并不严格要求结果的质量（如用于统计的事务），如果加上严格的封锁则是不必要和不经济的。因此有必要进行进一步的分析，考察不同级

别的一致性对数据库数据的质量及并行能力的影响。

【问题】 数据库事务的一致性级别是如何定义的？

【解析】 一致性级别定义为如下的几个条件。

① 事务不修改其他任何事务的脏数据（脏数据是被其他事务修改过，但尚未提交的数据）。

② 在事务结束前不对被修改的资源解锁。

③ 事务不读其他任何事务的脏数据。

④ 在读前对数据加共享锁（RS）和行排它锁，直至事务结束。

满足条件1的事务叫第0级事务。满足条件1和2的事务叫一级一致性事务。满足条件1、2和3的事务为二级一致性事务。Oracle的读一致性保证了事务不读其他事务的脏数据。满足条件1、2、3和4的事务叫三级一致性事务。

【例题1-41】 在SQL Server的安全体系中，登录对象和用户对象是SQL Server进行权限管理的两种不同的对象，这两种对象的区别是什么？

【解析】 在SQL Server中，登录对象和用户对象是SQL Server进行权限管理的两种不同的对象。一个登录对象是服务器方的一个实体，使用一个登录名可以与服务器上的所有数据库进行交互。用户这种对象是一个或多个登录对象在数据库中的映射，可以对用户对象进行授权，以便为登录对象提供对数据库的访问权限。用户获得一个登录名后，便可以立即在服务器上进行登录。但如果没有任何用户对象与之相对应，用户不能做任何操作。

每个数据库中都会有一个名为sysusers的表，这个表包含了在数据库中的所有用户对象，以及和它们相对应的登录名的标识。只有在master数据库中的syslogins表中，才会保存所有的登录名及口令。

所以，当一个登录名试图访问一个数据库时，SQL Server将在库中的sysusers表中查找对应的登录名。如果不能将登录名映射到数据库用户上，系统将试图将该登录名映射成guest用户（如果当前的数据库中有guest用户的话）。如果还是失败的话，这个用户将无法访问数据库。

【例题1-42】 阅读以下有关数据库的参照完整性的叙述，回答问题1和问题2。

【问题1】 为了维护数据库的参照完整性，当删除被参照关系的元组时，系统可能采取哪些做法？

【解析】 为了维护数据库的参照完整性，当删除被参照关系的元组时，系统可能采取如下3种做法。

① 级联删除（Cascades）：即当删除被参照关系的元组时，同时将参照关系中所有外键值与被参照关系中要被删除元组的主键值相等（相对应）的元组一起删除。

② 拒绝删除（Restricted）：即只当参照关系中没有任何元组的外键值与被参照关系中要被删除的元组的主键值相等（相对应）时，系统才执行该删除操作，否则拒绝执行该删除操作。

③ 置空值删除（Nullfies）：即当删除被参照关系的元组时，同时将参照关系中所有与被参照关系中要被删除元组的主健值相等（相对应）的外键值都置为空值。

【问题 2】 若有学生关系 S（S#，SNAME，SEX，AGE），其主键为 S#；选课关系 SC（S#，C#，GRADE），其主键为（S#，C#），且 S.S#=SC.S#。假定学生号为'01001'的学生离开学校不再回来了，为此若删除关系 S 中 S#='01001'的元组时，如果关系 SC 中有 4 个元组的 S#='01001'，应该选用哪一种做法？为什么？

【解析】 对于本题的情况，应该选用第一种做法。即将关系 SC 中 S#='01001'的 4 个元组也一起删除。

因为当一个学生离开学校不再回来，他的个人信息记录若从 S 关系中删除了，那么他的选课信息记录就没有保存的必要，也应随之从 SC 关系中删除。

【例题 1-43】 阅读以下关于 Oracle 系统方面的叙述，回答问题。

为了保护数据的完整性，数据库程序员用了很多的方法，比如数据表的主键约束、外键约束、触发器等等。在数据事务处理的时候，如何保存数据的完整性成为一个程序员经常遇到的问题。

【问题】 在设计一个 Oracle 数据库系统中，技术员遇到这样一个问题：在一个存储过程中 Proc_SaveBill 保存一张单据，调用一个函数 Func_GenerateCode()来生成一个单据的编号，这个函数是从一个存有最大编号的表中取出的，并且把编号加一。但是在 Proc_SaveBill 中的保存之前通过 Func_GenerateCode()得到一个编号，如果后面操作不成功，就会丢失一个编号，因为调用一次 Func_GenerateCode()就会增加一个编号的。为了使数据完整不至于丢单，技术员使用了事务回滚，但是又有问题了，因为函数 Func_GenerateCode()中已经有事务的 COMMIT 语句，也就是出现事务的嵌套，Proc_SaveBill 出错后还是不能将单号回滚到最初。请您尝试解决这个问题。

【解析】 为了解决这个问题，Oracle 中用自治事务来处理上述出现的问题，在存储过程的 is\as 后面声明 PRAGMA AUTONOMOUS_TRANSACTION；自治事务防止嵌套提交，使事务在自己的事务区内提交或回滚不会影响其他的事务。

【例题 1-44】 阅读以下关于电子售票系统方面的叙述，回答问题。

某电子售票系统的建设中，采用的都是 Sybase 数据库，操作系统为 SCO UNIX，在具体使用的过程中，发现了一些问题，干扰了日常工作的正常进行。

【问题】 Sybase 数据库启动后，无法正常使用，各种命令均不能使用。分析 Sybase 数据库日志文件/u/sybase/install/errorlog，它记载了所有有关数据库方面的出错信息，经检查其中有出错信息。从该信息中技术员断定数据库启动时无法打开 Sybsystemprocs 库，该库被标志为“挂起”。如果技术员判定信息无误，应如何解决该问题？

【解析】 Sybsystemprocs 库主要是存放系统的存储过程，Sybase 数据库提供的系统命令均在此库中，它的损坏必然导致数据库无法正常使用。首先修改 Sybase.cfg 文件，设置 Sybase 数据库允许修改系统参数，然后重新启动数据库，用 isql 登录到 SQL Server，修改

master 库的系统表 Sysdatabases 中对应 Sybsystemprocs 库的 Status 的值为–32768，通知 Sybase 强行启动 Sybsystemprocs 库，重新启动数据库正常后，再将 Sybase.cfg 文件中 Allow Updates 的值改为 0。这样 Sybsystemprocs 库被挂起的故障就解决了。

【例题 1-45】 阅读以下触发器方面的叙述，回答问题。

触发器是一种特殊的存储过程，它在插入，删除或修改特定表中的数据时触发执行，它比数据库本身标准的功能有更精细和更复杂的数据控制能力。

【问题】 根据自己的项目经验，列举触发器在数据库安全性、完整性的作用。

【解析】 数据库触发器有以下的作用。

① 安全性。可以基于数据库的值使用户具有操作数据库的某种权利。可以基于时间限制用户的操作，例如不允许下班后和节假日修改数据库数据。可以基于数据库中的数据限制用户的操作，例如不允许股票的价格的升幅一次超过 10%。

② 审计。可以跟踪用户对数据库的操作，审计用户操作数据库的语句，把用户对数据库的更新写入审计表。

③ 实现复杂的数据完整性规则。实现非标准的数据完整性检查和约束。触发器可产生比规则更为复杂的限制。与规则不同，触发器可以引用列或数据库对象，例如，触发器可回退任何企图吃进超过自己保证金的期货。提供可变的默认值。实现复杂的非标准的数据库相关完整性规则。触发器可以对数据库中相关的表进行连环更新，例如，在 auths 表 author_code 列上的删除触发器可导致相应删除在其他表中的与之匹配的行。在修改或删除时级联修改或删除其他表中的与之匹配的行。在修改或删除时把其他表中的与之匹配的行设成 NULL 值。在修改或删除时把其他表中的与之匹配的行级联设成默认值。触发器能够拒绝或回退那些破坏相关完整性的变化，取消试图进行数据更新的事务。当插入一个与其主健不匹配的外部键时，这种触发器会起作用，例如，可以在 books.author_code 列上生成一个插入触发器，如果新值与 auths.author_code 列中的某值不匹配时，插入被回退。

【例题 1-46】 阅读以下 Oracle 触发器方面的叙述，回答问题。

由于厂商不同，数据库的触发器的语法也有一定的区别，Oracle 产生数据库触发器的语法为：

```
CREATE [OR REPLACE] TRIGGER 触发器名 触发时间 触发事件
ON 表名
        [FOR EACH ROW]
        PL/SQL 语句
```

其中专有名词解释如下所述。

① 触发器名：触发器对象的名称。由于触发器是数据库自动执行的，因此该名称只是一个名称，没有实质的用途。

② 触发时间：指明触发器何时执行，该值可取 BEFORE——表示在数据库动作之前触发器执行；AFTER——表示在数据库动作之后出发器执行。

③ 触发事件：指明哪些数据库动作会触发此触发器；INSERT——数据库插入会触发此触发器；UPDATE——数据库修改会触发此触发器；DELETE——数据库删除会触发此触发器。

④ 表名：数据库触发器所在的表。

⑤ FOR EACH ROW：对表的每一行触发器执行一次。如果没有这一选项，则只对整个表执行一次。

【问题】 请写出触发器 auth_secure 在更新表 auths 之前触发，目的是不允许在周末修改表。

【解析】

```
CREATE TRIGGER auth_secure
      BEFORE INSERT OR UPDATE OR DELETE   //对整表更新前触发
      ON auths
    BEGIN
      IF（TO_CHAR(SYSDATE,'DY')='SUN'
        RAISE_APPLICATION_ERROR(–20600,'不能在周末修改表 auths');
      END IF;
    END
```

【例题 1-47】 阅读以下视图方面的叙述，回答问题。

视图是原始数据库数据的一种变换，是查看表中数据的另外一种方式。可以将视图看成是一个移动的窗口，通过它可以看到感兴趣的数据。

视图是从一个或多个实际表中获得的，这些表的数据存放在数据库中。那些用于产生视图的表叫做该视图的基表。一个视图也可以从另一个视图中产生。

视图的定义存在数据库中，与此定义相关的数据并没有再存一份于数据库中。通过视图看到的数据存放在基表中。

视图看上去非常像数据库的物理表，对它的操作同任何其他的表一样。当通过视图修改数据时，实际上是在改变基表中的数据；相反地，基表数据的改变也会自动反映在由基表产生的视图中。由于逻辑上的原因，有些视图可以修改对应的基表，有些则不能（仅仅能查询）。

【问题】 在数据库系统中，视图有哪些作用？

【解析】 简单性。看到的就是需要的。视图不仅可以简化用户对数据的理解，也可以简化他们的操作。那些被经常使用的查询可以被定义为视图，从而使得用户不必为以后的操作每次指定全部的条件。

安全性。通过视图用户只能查询和修改他们所能见到的数据。数据库中的其他数据则既看不见也取不到。数据库授权命令可以使每个用户对数据库的检索限制到特定的数据库对象上，但不能授权到数据库特定行和特定的列上。通过视图，用户可以被限制在数据的

不同子集上。

逻辑数据独立性。视图可帮助用户屏蔽真实表结构变化带来的影响。

【例题 1-48】 阅读以下关于人事信息系统数据库分析方面的叙述，回答问题 1 到问题 3。

某大型企业集团有 5 万多名人员分布在“总部”—省或直辖市级“公司”—县级市“子公司”三级，分别从事着生产、营销和管理等各类业务。为了充分发挥管理人才和专业人才等各类人员的特长，总部信息部门工程师根据总部办公会议的决定，已着手建立一个人事管理信息系统，用于收集、存储、加工、检索、分析和传输全集团范围内的人事信息资源。经集体讨论与分析，工程师提出了以下的一些想法。

① 人事管理信息系统的数据类型可以是相当丰富的，包括以下几种。

- ❑ 结构化数据——如人员与机构的基本信息，劳动工资与福利信息和专业与职务演变信息等。
- ❑ 非结构化数据——如照片、音像资料、地图文件、政策文件和有关的证书证明材料等。

建议采用以下 3 类服务器分别存放有关的信息与数据。

- ❑ Oracle 服务器——以数据库文件形式存放各类报表、人员基础数据、字典代码和信息分析数据等。
- ❑ Domain 服务器——以 Notes 库文件形式存放照片、政策文件、字典代码等。
- ❑ NT 服务器——以文件系统方式存放音像资料、地图、传送的报表、资源概览文件、名册和常用表格文件等。

② 整个集团人事数据库的容量不算太大，为了便于科学管理和减少日常维护的工作量，建议由总部统一开发与规划信息系统，采用总部—公司—子公司三级管理的方式设置数据库与文件系统。总部、每个公司和每个子公司都统一设有上述 3 类服务器，但存放内容的范围有所不同，即采用“逐级集中”的存放方式。

- ❑ 子公司只存放子公司及其所管辖的有关人员的数据。
- ❑ 公司存放有公司本身的数据，同时也存放它所管辖的那些子公司的数据。
- ❑ 总部存放有总部本身的数据，同时还存放所管辖的所有公司和所有子公司的数据。

就基础数据而言，总部和公司都分别有两个数据库，一个是本部库，另一个是所管辖的下级库。子公司只需要有一个数据库就可。

③ 该人事管理信息系统在初始建立时数据工作量相当大，然而日常处理的数据量不会很大，只需要限于发生变化的那些数据即可。大体上，人事信息系统的数据流程如下所述。

- ❑ 子公司收集本部门及其所管辖人员的所有数据到子公司当前人事库中。
- ❑ 公司收集本部的数据到公司当前本部人事库，并把所管辖的那些子公司的当前人事库数据汇总入总公司当前所辖的下级库。此外，需要把有关数据提取到该公司的“数据仓库”中。

- ❑ 类似地，总部收集本部数据至总部当前本部库，并把所有公司的当前两个库汇总入总部当前所辖库内。此外，还需定期地把相应的数据提取到总部的“数据仓库”中。事实上，总部或公司的数据仓库数据是按照分析和统计的主题进行组织的。

④ 人事信息系统所需的代码、字典与模板等可由总部→公司→子公司逐级向下分发。

⑤ 这类结构的人事信息系统实施时的一个关键问题是“数据的传输复制与分发”任务。建议采用以下 3 种方式进行。

- ❑ 网上数据复制方式——每月进行一次，即在网络中由下向上地传输变动的基本数据，照片库信息等；也可以由上向下地发送系统的字典代码、报表模板等。
- ❑ 数据邮递方式——需要随时进行，即采用 Notes 的邮件功能在网上传输报表、地图、人员控制数据、资源概览数据等，根据需要可以上报或下发。
- ❑ 手工磁带传递方式——包括系统在初始化时的系统初始数据以及音像资料变化时的音像文件。采用手工方式是因为这两类内容数据量太大，网络传输将会占用太多的带宽。

【问题 1】 在实现第一种网上数据复制方式进行基础数据定期上传时，工程师认为可以采用以下实现方式：下级库管理员预先把在日志中对 Oracle 数据库中有关库修改操作的 SQL 语句（如 insert、updata 或 delete）记录下来，作为 Notes 库的文档，用 Notes 复制技术上传至相应的上级部门。

【解析】 这些 SQL 语句纪录可作为系统的附加备份，当系统出现损害或破坏，可用这些 SQL 语句恢复系统数据。

【问题 2】 工程师还认为“数据邮递方式”应当设计有“函件封装”和“函件传输”两个模块，请用 100 字以内文字简要说明这两个模块的主要功能是什么？

【解析】 函件封装：提供给报表、地图、资源概览、人员控制等相应模块调用，实现传输信息的封装。函件传输：完成用户提交函件的发送、接收、延时重发和异常处理。

【问题 3】 请用 100 字以内文字简要分析在本系统中采用上述 3 类服务器工作方式的优缺点。

【解析】 优点：充分发挥不同服务器特长，日常运行效率高；统一规划，便于管理；灵活方便，易于扩展。

缺点：代价太高（特别是在子公司很多的场合下）。

【例题 1-49】 在关系数据库中，关系的完整性是指什么？主要包括哪几个方面？请解释这几个方面的实际意义。

【解析】 关系的完整性是指关系中数据值与其描述的应用对象实际状态保证一致的约束条件，主要包括域完整性、实体完整性、参照完整性和商务约束几个方面。

① 域完整性：域完整性规定了属性的值必须是域中的值，一个属性值能否为“空”由实际的应用语义决定。

② 实体完整性：指关系中的主关键字不能为空且主关键字的值不能相同，保证主关

键字能唯一地标识关系中的每个元组。

③ 参照完整性：指不允许引用数据库中不存在的外键数据，外键（或叫外部关键字）是指一个表中的某个属性是另一个表的主关键字。

【例题 1-50】 某单位数据库环境为 Oracle 7.3，开发工具是 Develope 2000。请举例说明 Oracle 7.x 中账户安全（Account Security）、系统级权限（System-Level Privilege）、对象安全性（Object Security）、审计（Auditing）的含义。

【解析】 账户安全：要在数据库中访问数据，就必须访问该数据库的一个账户。每个账户必须指定一个口令。口令是在账户建立的时候设置的，可由 DBA 或用户进行修改。

系统级权限：系统级权限可以建立从系统级权限全集到扩展的基本系统级的各类角色。比如 Connect、Re source 和 DBA 就是分别提供给用户、开发者及 DBA 的标准角色。

对象安全性：用户可以通过 GRANT 命令将自己创建的一些权限授予其他用户使用，也可以给其他用户授予对对象授权的权限。例如，可以授予一个用户拥有对本用户表授予 SELECT 权限的权限。

审计：Oracle 具有审计发生在其内部的所有操作，包括注册企图，对象访问和数据库操作的能力。审计的结果存储在数据库的审计表中。

【例题 1-51】 在 Oracle 7.x 系统中，计算机发生故障有多种形式，包括数据文件损坏、控制文件损坏、整个文件系统损坏等，根据自己的项目经验谈谈怎样恢复该系统？

【解析】 数据文件损坏时可以用最近所做的数据库文件备份进行恢复，即将备份中的对应文件恢复到原来位置，重新加载数据库；控制文件损坏时，若数据库系统中的控制文件损坏，则数据库系统将不能正常运行，那么，只须将数据库系统关闭，然后从备份中将相应的控制文件恢复到原位置，重新启动数据库系统；整个文件系统损坏时，在大型的操作系统中，如 UNIX，由于磁盘或磁盘阵列的介质不可靠或损坏是经常发生的，这将导致整个 Oracle 数据库系统崩溃，这种情形只能按照以下步骤进行：将磁盘或磁盘阵列重新初始化，去掉失效或不可靠的坏块；重新创建文件系统；利用备份将数据库系统完整地恢复；启动数据库系统。

【例题 1-52】 分别叙述静态转储和动态转储？

【解析】 静态转储：指转储期间不允许（或不存在）对数据库进行任何存取、修改活动。静态转储简单，但转储必须等待用户事务结束才能进行，同样，新的事务必须等待转储结束才能执行。显然，这会降低数据库的可用性。

动态转储：指转储期间允许对数据库进行存取或修改，即转储和用户事务可以并发执行。动态转储可克服静态转储的缺点，但是，转储结束时后原副本上的数据并不能保证正确有效。

【例题 1-53】 什么叫做事务？事务有哪几个基本特征？

【解析】 事务是指一个单元的工作，这些工作要么全做，要么全部不做。作为一个逻辑单元，必须具备 4 个属性：自动性、一致性、独立性和持久性。

自动性是指事务必须是一个自动的单元工作，要么执行全部数据的修改，要么全部数据的修改都不执行。

一致性是指当事务完成时，必须使所有数据都具有一致的状态。在关系型数据库中，所有的规则必须应用到事务的修改上，以便维护所有数据的完整性。所有的内部数据结构，例如树状的索引与数据之间的链接，在事务结束之后，必须保证正确。

独立性是指并行事务的修改必须与其他并行事务的修改相互独立。一个事务看到的数据要么是另外一个事务修改这些事务之前的状态，要么是第二个事务已经修改完成的数据，但是这个事务不能看到正在修改的数据。这种特征也称为串行性。

持久性是指当一个事务完成之后，它的影响永久性地产生在系统中，也就是这种修改写到了数据库中。

【例题 1-54】 某电子一卡通系统的建设中，采用的都是 SQL Server 2000 数据库，操作系统为 Windows 2000 Server，技术员在安装时发现，SQL Server 2000 在管理系统安全性上采用了两种安全认证模式：Windows NT 认证模式和混合认证模式。那二者究竟在实际应用过程中以及应用范围上有何不同呢？应如何进行安全模式的设置呢？

【解析】 SQL Server 通过选择不同的安全认证模式，对用户进行相应的登录验证，它既可用 Windows NT 为用户提供的账号验证用户的合法性，又可利用其自身为用户提供的登号库校验用户连接的有效性。

SQL Server 在这两种模式下对登录用户的验证过程不同，而且各有不同的适用范围。当采用基于 Windows NT 的认证机制时，客户在连接 SQL Server 时，同时也就和服务器端建立了一种可信连接，这个连接可以将该用户的所在工作组名、用户账号传递到 SQL Server。由于此时为可信连接，SQL Server 也就会认为该用户是已经通过了 Windows NT 的校验并认为是合法用户，因此 SQL Server 只须查找它自己系统登录表（Syslogins），如果该用户在表中，则认为此次连接合法通过，而不再验证用户的口令；如果该用户不在登录中，则拒绝此次连接请求。这种安全模式比较适用于在一个域或一组有信任关系的域中存在多个 SQL Server 服务器的情况，它使得用户在一个域服务器上登录后，即可访问该域中以及其信任域中的任何 SQL Server 服务器。而且，这种模式可以使 SQL Server 充分利用许多 Windows NT 在安全性管理上的优势，加强安全防范。例如：Windows NT 的访问审计功能，对于用户非法口令的锁定，用户口令有效期管理，口令加密、限制长度等功能。当采用混合认证模式时，系统首先检验一下用户是否为 Windows NT 用户，如果成立，则接下来的认证过程同前；否则，SQL Server 再检验该用户是否在其系统登录表中，如存在该账号，则允许用户连接，否则断开连接。这种安全模式的优势在于它可以使非 Windows NT 客户，Internet 客户、以及混合客户组连接到 SQL Server 上，而且实际的安全性是在 Windows NT 的安全层上又多了一层屏障。

由于这两种模式各有优势，在进行安全模式设置时，应综合多方面因素，权衡利弊。设置安全模式的过程分以下步骤。

① 验证一下当前的连接是否为可信连接；利用 SQL Server 的企业管理器设置相应的安全模式；

② 停止并重新启动一下服务以使设置生效；

③ 创建在可信连接中用于登录 SQL Server 服务器的 Windows NT 用户或工作组，此项工作须由有管理员权限的人员来完成。

④ 利用 SQL Servre 的企业管理器授予以上新建用户访问 Sql Server 的权力。

⑤ 设置其他不通过可信连接来访问 SQL Server 的用户的访问权，为他们创建 SQL Server 的登录账号和口令以及默认语言。

1.5　编写外部设计文档

系统文档是是开发人员与用户、开发人员之间交流的工具，是系统测试人员和系统维护人员的指南。本节主要讲述编写系统说明书的要求，包括系统配置图、各子系统关系图、系统流程图，系统功能说明、输入输出规格说明、数据规格说明、用户手册框架等。如图 1-20 所示是本节的知识框图。

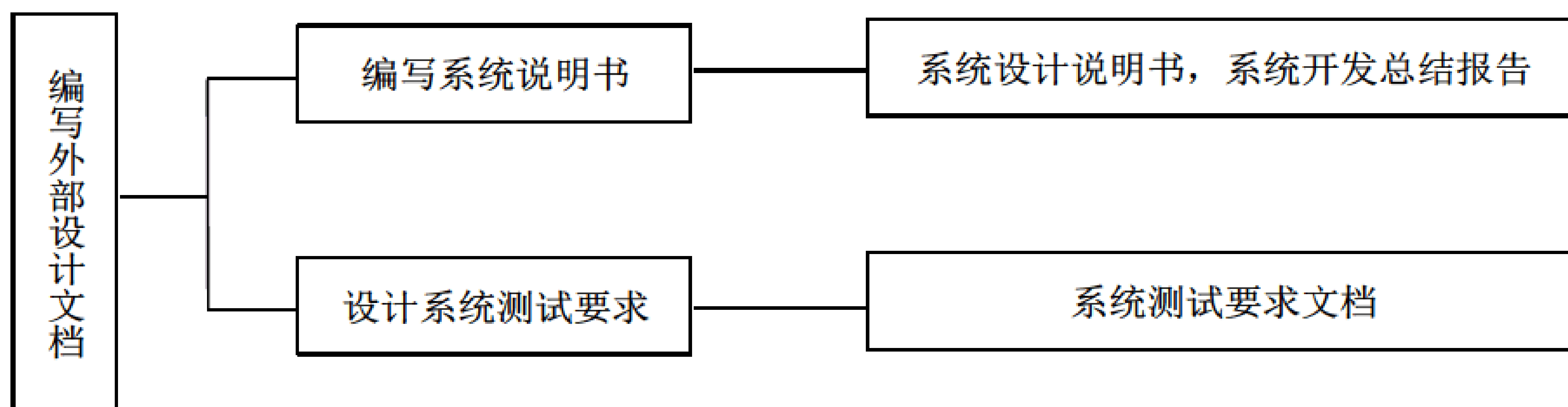


图 1-20　编写外部设计文档知识框图

1．知识点提炼

（1）编写系统说明书（系统配置图、各子系统关系图、系统流程图，系统功能说明、输入输出规格说明、数据规格说明、用户手册框架）

系统文档是系统维护人员的指南，是开发人员与用户交流的工具。规范的文档是规范化开发的必要组成部分，是系统质量的保障。系统文档主要有系统设计说明书和系统开发总结报告。开发总结报告分为研制报告、技术报告和技术手册 3 个文档，其中的技术手册记录了系统开发过程中的各种主要技术细节。这样，即使系统维护人员不是原来的开发人员，也可以在这些文档的基础上进行系统地维护与升级。

用户与维修人员在运行维护期间进行沟通。用户在使用信息系统过程中，将运行过程中的问题进行记载，形成系统运行报告和维修修改建议。系统维护人员根据维护修改建议以及系统开发人员留下的技术手册等文档，对系统进行维护和升级。

（2）设计系统测试要求

作为数据库系统开发的重要环节，系统的测试日益受到重视。随着系统开发规模的增大、复杂程度的增加，测试工作就显得更加困难。系统测试的依据是规格说明书、设计文档和使用说明书，如果设计有错误，测试的质量就难以保证。为了保证系统的质量，在开发过程中，必须对各阶段形成的结果，分别进行严格的审查。因为系统的测试要求与用户需求有着密切的关系，所以在数据库设计阶段就要设计出系统的测试要求，而不是系统基本完成之后才设计测试要求。同时，系统的测试必须规范化，并编写测试文档，以文档的形式将设计的测试要求保存下来，作为以后系统测试的主要参考。

系统测试要求文档中要详细规定测试的要求，包括测试的目的和内容、方法、步骤以及测试的各项指标等。通常，系统测试要求贯穿于数据库设计的整个阶段，测试文档的编写需要从需求分析阶段开始，到系统设计阶段结束时完成。

2．难点分析

文档在系统开发人员、项目管理人员、系统维护人员、系统评价人员以及用户之间的多种作用总结如下。

用户与系统分析人员在系统规划和系统分析阶段通过文档进行沟通。这里的文档主要包括可行性研究报告、总体规划报告、系统开发合同、系统方案说明书等。有了文档，用户就能依次对系统分析员是否正确理解了系统的需求进行评价，如不正确，可以在已有文档的基础上进行修正。

系统开发人员与项目管理人员通过文档在项目期内进行沟通。这里的文档主要有系统开发计划（包括工作任务分解表、网络图、甘特图、预算分配表等）、系统开发月报以及系统开发总结报告等项目管理文件。有了这些文档，不同阶段之间的开发人员就可以进行工作的顺利衔接，同时还能降低因为人员流动带来的风险，因为接替人员可以根据文档理解前面人员的设计或开发思路。

系统测试人员与系统开发人员通过文档进行沟通。系统测试人员可以根据系统方案说明书、系统开发合同、系统设计说明书、测试计划等文档对系统开发人员所开发的系统进行测试。系统测试人员再将评估结果撰写成系统测试报告。

系统开发人员与用户在系统运行期间进行沟通。用户通过系统开发人员撰写的文档运行系统。这里的文档主要是用户手册和操作指南。

系统开发人员与系统维护人员通过文档进行沟通。

系统测试要求文档的重要性主要表现在以下几个方面。

① 验证系统需求的正确性。测试文档中规定了用以验证系统需求的测试条件，研究这些测试条件对弄清用户的需求是十分有帮助的。

② 检验测试资源。测试要求不仅要用文件的形式把测试过程规定下来，还需要说明测试工作必不可少的资源，进而检验这些资源是否可以得到，即它的可用性如何。

③ 生成测试用例。测试用例的好坏决定着测试工作的效率，选择合适的测试用例是

做好测试工作的关键。在测试文件编制过程中，按规定的要求精心设计测试用例有重要的意义。

④ 评价测试结果。测试文件包括测试用例，即若干测试数据及对应的预期测试结果。完成测试后，将测试结果与预期的结果进行比较，便可对已进行的测试提出评价意见。

⑤ 再测试。测试文件规定的和说明的内容对维护阶段由于各种原因的需求进行再测试时，是非常有用的。

1.6　设计评审

在数据库开发和投入使用后，都需要对数据库进行评审。通过多种方式监测数据库的性能指标，得到评审结果，对不满足评审指标的数据库应采取相应的措施，如数据库的重组织，甚至数据库或者应用系统的重新设计与重构造。本节主要讲述了信息系统评审的基本概念和基本指标。如图 1-21 所示是本节的知识框图。

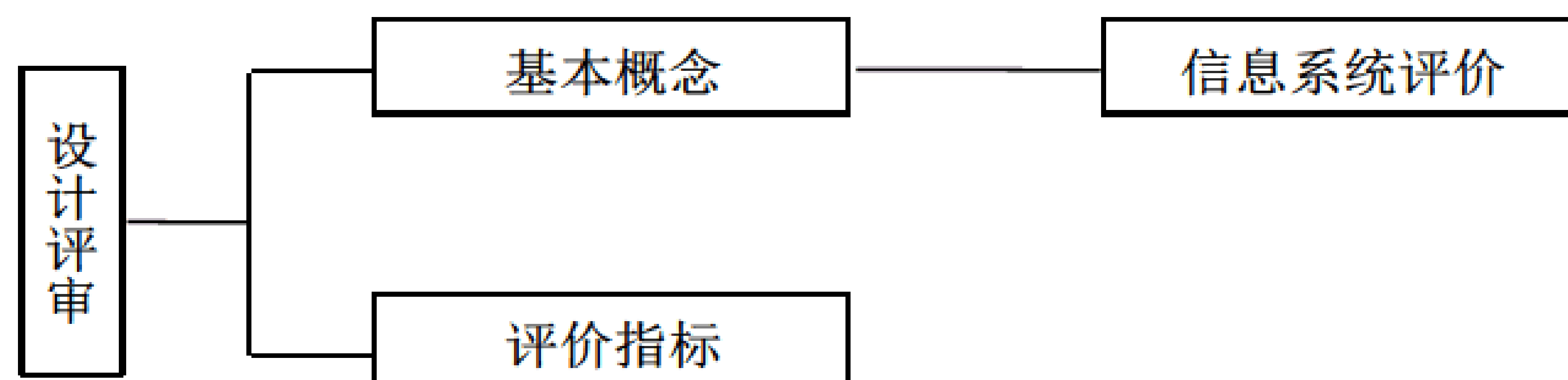


图 1-21　数据库设计知识框图

1．知识点提炼

信息系统的评价分为广义和狭义两种。广义的信息系统评价是指从系统开发的一开始到结束的每一阶段都需要进行评价。狭义的信息系统评价则是指在系统建成并投入运行之后所进行的全面、综合的评价。

按评价的时间与信息系统所处的阶段的关系，又可从总体上把广义的信息系统评价分成立项评价、中期评价和结项评价。

（1）立项评价

立项评价指信息系统方案在系统开发前的预评价，即系统规划阶段中的可行性研究。评价的目的是决定是否立项进行开发，评价的内容是分析当前开发新系统的条件是否具备，明确新系统目标实现的重要性和可能性，主要包括技术上的可行性、经济上的可行性、管理上的可行性和开发环境的可行性等方面。由于事前评价所用的参数大都是不确定的，所以评价的结论具有一定的风险性。

（2）中期评价

项目中期评价包含两种含义，一是指项目方案在实施过程中，因外部环境出现重大变化，比如市场需求变化、竞争性技术或更完美的替代系统的出现，或者发现原先设计有重

大失误等，需要对项目的方案进行重新评估，以决定是继续执行还是终止该方案；另一种含义也可称为阶段评估，是指在信息系统开发正常情况下，对系统设计、系统分析、系统实施阶段的阶段性成果进行评估，由于一般都将阶段性成果的提交视为信息系统建设的里程碑，所以，阶段评估又可叫里程碑式评价。

（3）结项评价

信息系统的建设是一个项目，是项目就需要有终结时间。结项评价是指项目准备结束时对系统的评价，一般是指在信息系统投入正式运行以后，为了了解系统是否达到预期的目的和要求而对系统运行的实际效果进行的综合评价。所以，结项评价又是狭义的信息系统评价。信息系统项目的鉴定是结项评价的一种正规的形式。结项评价的主要内容包括系统性能评价、系统的经济效益评价以及企业管理效率提高、管理水平改善、管理人员劳动强度减轻等间接效果。通过结项评价，用户可以了解系统的质量和效果，检查系统是否符合预期的目的和要求；开发人员可以总结开发工作的经验、教训，这对今后的工作十分有益。

系统评价指标包括：系统质量、技术水平、运行质量、用户需求、系统成本、系统效益和财务评价。

2．典型例题

【例题 1-55】 阅读以下复审、复查、管理复审和测试有关的叙述，回答问题 1 到问题 3。

软件产品生产周期长、耗资巨大，必须特别注意保证质量，而通常保证软件质量的措施可归为 4 方面，即复审、复查、管理复审和测试，不同的方面反映了软件质量保证措施中的不同需要，试回答以下问题并适当加以阐述。

【问题 1】 复审、复查、管理复审和测试各自包括的具体内容是什么，它在哪些方面对软件质量的保证产生了作用？

【解析】 本题主要考查考生对软件质量保证措施的全面了解程度。软件质量保证措施除了常规的对编码进行测试外，还有软件生命周期阶段成果的复审，对各阶段文档和材料的复查，从管理的角度对软件开发工作的复审以及对编码的测试，要求学生建立软件质量保证的整体概念。

① 复审是在编码以前对分析、设计成果的非正规和正规审查，其重点是发现系统性的错误或缺点。

② 复查是指对阶段产生的文档和材料的检查，以保证下阶段工作的开始。

③ 管理复查是指从项目管理的角度，从总体、成本和进度等方面进行检查。

④ 测试是指对编码的查错和排错，应说测试的内容和过程，如单元测试、集成测试、系统测试等。

【问题 2】 软件复审和软件测试之间有什么联系，又有什么差别？各自有什么侧重？

【解析】 软件复审是软件在编码前对分析文档和设计文档的审查，其目的是发展和纠正在分析和设计阶段中可能产生的系统性错误，它是软件测试固有的重要内容和步骤。软

件测试是对基于正确设计基础上所开发的程序的测试。二者是从不同方面对软件质量的保证。

【问题 3】 软件测试的目的是什么，对其具体的内容和实现过程做一扼要陈述，无须对测试方法做出介绍。

【解析】 软件测试的目的从编码阶段来说是发现程序中可能出现的错误并排除错误。测试的具体内容是从不同范围和对象中来发现可能存在的错误并排除之，包括：单元测试即对模块进行测试，再对由模块集成的子系统进行测试，再将子系统集成起来进行系统测试，测试中将应用到测试实例和测试数据。

练习题

1．阅读以下关于软件维护方面的叙述，回答问题 1 和问题 2。

某企业主要从事于新型建筑材料的生产与销售，涉及到的产品的品种繁多，规格也不断推陈出新。该企业正在运行的生产与销售的几个管理软件是委托外地的某个软件公司完成的。在开始投入运行时，这几个软件的功能大体上能满足企业当时的实际工作状况。

由于企业的生产不断发展，企业决定扩大信息系统开发部的规模，以适应企业生产的发展需要。工程师带领开发部门的技术人员和管理人员一起，进行了软件运行情况的分析，向企业领导部门提出了下列主要意见。

① 由于在委托开发时，企业缺乏必要的经验，这几个软件虽然已提供了程序源代码，但其他文档相对而言过于简略。

② 如果需要再次委托原来的外地软件公司进行改进与维护，由于此外地软件公司的有关技术人员或者已跳槽，或者已在软件公司中担任繁重的管理工作，所需的维护代价极为昂贵。

③ 正在运行的一个主要管理软件，看来需要重新设计、重新编码和测试，因此，有必要先深入地去理解当前的设计。

④ 有两个辅助软件，要重点进行修改，以适应企业需求的当前变动与将来发展的可能变动情况。

【问题 1】 考虑到重新开发这个已在运行的主要管理软件，所需的代价相对而言比较高，企业领导与项目工程师一起进一步认真分析了该主要管理软件的运行情况，一致认为：

① 该主要管理软件的实际运行情况看来比较成功，在今后若干年中，企业确实需要依靠一个重新开发的主要管理软件，但功能上要作较多的扩充。

② 如果对原有的管理软件，进行不断地“修改”，预计每维护一行代码的代价可能会是原来开发该行代码代价的 15～50 倍，因此，渐进地加以修改看来很不适宜。

③ 企业已有实际使用经验，因此，可以相对容易地确定新的要求和修改的方向。

④ 原来的管理软件事实上已是一个原型，重新开发工作的生产效率也许可以更高

一些。

⑤ 在重新开发时，允许采用更好的设计概念去设计软件的结构，将会有助于今后的长期维护。

工程师还向企业领导提出，该管理软件的重新开发的本质是进行预防性的维护，使之适应于不断变化的企业需求，可以采用一些必要的工具使其中的某些工作得以自动完成。请用 120 字以内的文字简要说明这些工具的类型及其主要作用。

【问题 2】 对于因需求变更而引起修改的两个辅助软件来说，必须考虑到软件的修改十分费时，并且容易引进新的错误。工程师估计因需求变更，有 40 处需要变动，可以采用下列两种方案进行。

方案一：采用 36 处软件数据输入域变更和 4 处软件变更的方式。

方案二：采用 20 处软件数据输入域变更和 20 处软件变更的方式。

令软件的可修改度为 M，则 $M=M_r/M_s$，其中 M_r 为需求变更的数目，M_s 为软件修改处的数目。

请分别计算出方案一和方案二的可修改度，并说明哪一个方案更好，为什么？

2．阅读以下数据字典相关方面的叙述，回答问题 1 到问题 5。

对数据库设计来讲，数据字典是进行数据收集和数据分析所获得的主要成果。数据字典是各类数据描述的集合。数据字典通常包括数据项、数据结构、数据流、数据存储和处理过程 5 个部分。

【问题 1】 什么是数据项？对数据项的描述通常包括什么样的内容？

【问题 2】 什么是数据结构？对数据结构的描述通常包括什么样的内容？

【问题 3】 什么是数据流？对数据流的描述通常包括什么样的内容？

【问题 4】 什么是数据存储？对数据存储的描述通常包括什么样的内容？

【问题 5】 什么是处理过程？对处理过程的描述通常包括什么样的内容？

3．日常工作中，数据库的备份是数据库管理员必须不断进行的一项工作，Oracle 7 数据库的备份分为逻辑备份和物理备份两种。物理备份可以对 Oracle 数据库的所有内容进行复制，方式可以是多种，有脱机备份和联机备份，每种备份的特点是什么？

4．数据库设计是指什么？它有何特点？根据自己的项目经验，谈谈在数据库设计时应该注意一些什么样的问题？

5．什么是事务，事务的特点是什么？为什么很多时候没有显示提交事务，数据库也接 6 了用户的请求？

6．在关系数据库中，选择存取路径主要是指确定如何建立索引。根据自己的项目经验，谈谈建立索引的好处有哪些？一般来说，创建索引的原则有哪些？在什么情况下不适合建立索引？什么是聚簇索引和非聚簇索引？各适合什么样的应用情况？

7．在数据库系统中，锁的粒度并不是单一的，而是分层次的，这体现了并发度与加锁开销之间的一种平衡。一个比较典型的层次由下至上可能包括：属性、记录、页面、分

区（或表）、表（或表空间）、数据库。在这个层次中，每个粒度都是包含它的下一粒度的。采用多粒度的锁方案中，什么是锁升级（Lock Esclation）？采用锁升级，锁的粒度提高后会带来什么消极后果？请举例说明。

8．根据自己的项目经验，在理解系统需求说明时，谈谈数据库环境主要考虑哪些方面？

9．在数据库设计中，模式、外模式、内模式分别指什么？

10．根据自己的项目经验，列举设计成功的数据库系统所具备的特点？

11．阅读以下有关订单管理系统的叙述，回答问题1和问题2。

在一个订单管理系统中，订单实体如下：订单（订单号，订货日期，客户名称，厂商名称，厂商地址，产品名称，数量，价格，折扣）。

语义如下：每一个订单只对应一个客户，客户的名称是唯一的，每个订单只能订购一个厂商的产品，厂商名称是唯一的且只有一个地址，每个产品只能由一个厂商所生产，但一个厂商可以生产多个产品。不同的产品具有不同的价格，厂商根据所订购的产品和数量决定折扣。要求：实体分解的过程中只能使用原有的属性，不要产生如“厂商编号”一类的属性。

【问题1】 请给出该订单的函数依赖。

【问题2】 将订单实体规范化成3NF的关系。

12．阅读以下有合同管理系统的叙述，回答问题。

在一个合同管理系统中，公司的管理人员、销售人员、技术人员、财务人员需要查询合同中不同的内容。合同所包含的信息如下所述。

合同编号（NO）、销售人员（SALE）、客户信息（CUSTOMER）、合同总价（PRICE）、付款条件（PAY）、产品配置（PDT_CONFIG）、各产品价格（PDT_PRICE）和产品折扣（DISCOUNT）、服务条款（SERVICE）等，形式化描述如下：

CONTRACT（NO，SALE，CUSTOMER，PRICE，PAY，PDT_CONFIG，PDT_PRICE，DISCOUNT，SERVICE）

管理人员可以查询所有销售合同的所有信息；

销售人员可以查询自己的销售合同的所有信息；

技术人员可以查询销售人员、客户信息、产品配置和服务条款；

财务人员可以查看销售人员、客户信息、合同总价、付款条件。

【问题】 根据以上的需求，使用SQL视图定义设计出相应的视图。（使用规范的视图命名规则。）

参考：视图定义示例

```
CREATE   VIEW   V_EXAMPLE
AS SELECT NO，SALE，CUSTOMER，PRICE FROM CONTRACT
WHERE SALE="销售人员姓名";
```

13．阅读以下有关订单管理系统的叙述，回答问题 1 和问题 2。

一个图书借阅管理数据库要求提供下述服务。

随时查阅书库中现有书籍的品种、数量与存放位置。所有各类书籍均可由唯一书号标识。

可随时查询书籍借还情况。包括借书人单位、姓名、借书证号、借书日期和还书日期。（约定：任何人可借多种书，任何一种书可以为多个人所借，借书证号具有唯一性。）

当需要时，可通过数据库中保存的出版社的电报编号、电话、邮编以及地址等信息向有关书籍的出版社增购有关书籍。约定，一个出版社可以出版多种书籍，同一本书仅为一个出版社出版，出版社名具有唯一性。

【问题 1】 根据以上情况和假设，试做如下设计：构造满足需求的 E-R 图。

【问题 2】 将 E-R 图转换为等价的关系模式。

14．假设关系模式为 R（A，B，C，D），函数依赖为{A→B，B→C，B→D}。求：R 的所有键码。

15．在关系型数据库中，什么叫做封锁？一般来说，DBMS 提供的基本封锁类型有两种：排它锁（X 锁）和共享锁（S 锁），这两种锁对数据的读写有什么影响？根据自己的项目经验，谈谈锁的粒度对系统的并发程度和系统开销有什么影响？

16．什么是多值依赖中的数据依赖？举例说明。

17．数据库系统生存期是什么？

18．为什么说需求分析是数据库系统开发中最困难的任务之一？

19．简述 ORDBS 的中文含义。

20．数据库的三级模式和两级映像体系结构中，模式/内模式映像存在于概念级和内部级之间，用于定义概念模式和内模式间的对应性。其主要作用是什么？

21．简述逻辑数据的独立性。

22．数据库是一个共享资源，在多用户共享系统中，并发操作的含义是什么？

23．设有描述学校情况的 U 关系。

U（S#，SD，MN，CN，G）

其中：S＃属性表示学生学号，SD 表示学生所在系名，MN 表示系主任，CN 表示课程名，G 表示成绩。一个系有若干名学生，一个学生只属于一个系，一个系只有一名系主任，一个学生可选多门课，每个学生选每门课有一个成绩。试写出并关系中的函数依赖，并给每个函数依赖的一个简短说明。

24．什么是数据库的并发控制？

25．分别叙述数据与程序的物理独立性和逻辑独立性？

26．简述封锁类型的控制方式。

27．设有关系 S、SC、C，试用关系代数表达式完成下列操作。

S（snum，sname，age，sex）例：（001，'李强'，23，'男'）

SC（snum，cnum，score）　例：（003，'C1'，83）

C（cnum，cname，teacher）例：（'C1'，'数据库原理'，'王华'）

【问题 1】 检索“李强”同学不学的课程的课程号。

【问题 2】 检索学修了“李文”老师所授课程之一的学生的姓名。

28．设有关系 S、SC、C，试用关系元组演算表达式完成下列操作。

S（snum，sname，age，sex）例：（001，'李强'，23，'男'）

SC（snum，cnum，score）　例：（003，'C1'，83）

C（cnum，cname，teacher）例：（'C1'，'数据库原理'，'王华'）

【问题 1】 检索年龄大于 21 的女学生的学号和姓名。

【问题 2】 检索选修了“数据库系统概论”课程的学生姓名。

29．设关系模式 R（A，B，C，D，E，I），其函数依赖集为：{A→D，AB→E，BI→E，CD→I，E→C}，计算：{A，E}+，{A，E}是键码。

30．简述排它锁和共享锁的概念。

31．简述关系的完整性。

32．实体之间的联系有哪几种，并简述之。

33．根据自己的项目经验，列举在同用户商讨系统需求时，应注意哪些问题?

练习题答案

1．**【解析 1】**预防性维护可采用“逆向工程工具”和“再次工程工具”等 CASE 工具。前者主要用于分析已开发好的现有程序，获得程序的静态结构或动态性模型等。后者主要用于辅助代码重构、数据库重构与程序结构的改进等。

【解析2】$M_1 = M_r / M_s = 40/4 = 10$，$M_2 = M_r / M_s = 40/20 = 2$。由此可见方案一更好一些，因为数据输入域的变更修改会更加容易与更加清晰，应力求避免去修改软件本身。

2．**【解析 1】** 数据项是不可再分的数据单位。

数据项描述=｛数据项名，数据项含义说明，别名，数据类型，长度，取值范围，取值含义，与其他数据项的逻辑关系｝

其中取值范围、与其他数据项的逻辑关系定义了数据的完整性约束条件，是设计数据检验功能的依据。

【解析 2】 数据结构反映了数据之间的组合关系。一个数据结构可以由若干个数据项组成，也可以由若干个数据结构组成，或由若干个数据项和数据结构混合组成。

数据结构描述=｛数据结构名，含义说明，组成：｛数据项或数据结构｝｝

【解析 3】 数据流是数据结构在系统内传输的路径。

数据流描述=｛数据流名，说明，数据流来源，数据流去向，组成：｛数据结构｝，平均流量，高峰期流量｝

其中数据流来源是说明该数据流来自哪个过程。数据流去向是说明该数据流将到哪个

过程去。平均流量是指在单位时间（每天、每周、每月等）里的传输次数。高峰期流量则是指在高峰时期的数据流量。

【解析 4】 数据存储是数据结构停留或保存的地方，也是数据流的来源和去向之一。

数据存储描述=｛数据存储名，说明，编号，流入的数据流，流出的数据流，

组成：｛数据结构｝，数据量，存取方式｝

其中数据量是指每次存取多少数据，每天（或每小时、每周等）存取几次等信息。存取方法包括是批处理，还是联机处理；是检索还是更新；是顺序检索还是随机检索等。另外，流入的数据流要指出其来源，流出的数据流要指出其去向。

【解析 5】 数据字典中只需要描述处理过程的说明性信息，通常包括以下内容：

处理过程描述=｛处理过程名，说明，输入：｛数据流｝，输出：｛数据流｝，处理：｛简要说明｝｝

其中简要说明中主要说明该处理过程的功能及处理要求。功能是指该处理过程用来做什么（而不是怎么做），处理要求包括处理频度要求，如单位时间里处理多少事务，多少数据量；响应时间要求等。这些处理要求是后面物理设计的输入及性能评价的标准。

3.**【解析】** 脱机备份是在 Oracle 数据库正常关闭后，对 Oracle 数据库进行备份，备份的内容包括：所有用户的数据库文件和表、所有控制文件、所有的日志文件、数据库初始化文件等。可采取不同的备份方式，如：利用磁带转储命令（TAR）将所有文件转储到磁带上，或将所有文件原样复制（COPY，RCP）到另一个备份磁盘中或另一个主机的磁盘中。联机备份可以将联机日志转储归档，在 Oracle 数据库内部建立一个所有进程和作业的详细准确的完全记录。

4.**【解析】** 定义：数据库设计是指在给定的应用环境下，根据用户的应用需求构造优良的数据库模式，建立数据库及其应用系统，使之能够有效地存储数据，满足各种用户的应用需求。

特点：数据库设计侧重于对数据的分析和设计，数据库设计应该在整个设计过程中把结构（数据）设计和行为（处理）设计密切结合起来。

设计时特别注意以下问题。

① 按照数据库的生命周期，分阶段进行开发和管理。

② 在设计中遵守结构化程序设计的原则：自顶向下、逐步求精、分而治之。

③ 在每个阶段都应建立适当的文档。

④ 加强数据库设计的管理，人员要合理组织并有层次，设计进度应有计划并加以控制，开发成本要有估计。

⑤ 设计时要着眼于整个生命周期，考虑当前和今后的发展，并使使用和维护方便。

⑥ 重视数据库设计的标准化和规范化。

5.**【解析】** 事务，就是一个完整的活动序列，它包含一组操作，这些操作或者全部成功地执行，或者都不执行并恢复到执行前的状态。

事务处理有以下 4 个特点（即所谓的 ACID）。

① 原子性（Atomicity）：事务是一个活动的逻辑单元，该逻辑单元的工作要么全部被提交，要么全部被消。

② 一致性（Consistency）：事务是一个完整的逻辑单元，能够把共享资源（比如一个数据库的表）从一个有效状态变迁到另一个有效状态。

③ 独立性（Isolation）：事务在对共享资源的修改被提交之前，在该事务之外是看不到的。

④ 持久性（Durability）：事务对共享资源的修改即使是在系统或存储介质出现故障时，也能被保存下来。

关系数据库系统通过日志实现了事务处理。在关系数据库系统中，一般把事务分为两类：隐含事务和显式事务。系统将自动提交隐含事务，而对于显式事务，用户必须使用事务提交命令（一般是 COMMIT）显式地完成事务的提交。在执行 CREATE、DROP、GRANT、REVOKE 等命令时，系统将自动完成事务的提交。

下面以 Oracle 为例说明。Oracle 自动提供行级锁，它允许用户在没有冲突的情况下更新表中不同的行。行级锁对联机事务处理非常有用。在正常情况下，Oracle 会自动锁住需要加锁的资源以保护数据，这种锁是隐含的，叫隐含锁。然而，在一些条件下，这些自动的锁在实际应用时并不能满足需要，必须人工加一些锁。这些人工加的锁叫显式锁。

下面指明了会产生隐含锁的 SQL 语句：INSERT、UPDATE、DELETE、DDL/DCL 语句。

下面指明了会产生显式锁的 SQL 语句：SELECT FOR UPDATE；LOCK TABLE IN XXX MODE。

解决读的不可重复性可以用下面的方法。在 Oracle 中，用 SELECT FOR UPDATE 对预期要修改的记录加行排它锁（X），对表加行共享锁（RS）。它常用于要锁住一行，但不去真的修改这一行。

6.【解析】 建立索引有以下好处。

- 可以加快数据的检索速度。
- 可以加速表与表之间的连接。
- 创建唯一性索引可以保证数据的唯一性。
- 在使用 ORDER BY 或 GROUP BY 进行搜索时，可以减少排序和分组的时间。

一般来说，考虑在如下的字段上创建索引。

- 经常搜索的列。
- 在主键上。
- 在外键上。
- 根据范围搜索的列上。
- 要经常排序的列上。
- 经常使用 WHERE 子句的列上。

一般不在以下的字段上创建索引。

- 只有很少值的列，如性别，真假。
- 大文本、图像字段。
- 查询中很少使用的列。

聚簇索引和非聚簇索引概念：聚簇索引是行的物理顺序和索引顺序相同，也就是说，聚簇索引的叶子是实际的数据页；非聚簇索引行的物理顺序和索引顺序不同，非聚簇索引的叶子是指向实际数据页的指针。一个表只能有一个聚簇索引，但可以有多个非聚簇索引。创建非聚簇索引之前要创建聚簇索引。

在以下情况用聚簇索引。

- 查询的字段返回大的结果集，考虑为该字段加聚簇索引。
- 含有有限（不很少）数目唯一值的字段。
- 表中经常搜索的列或者按照顺序访问的列。

以下情况用非聚簇索引。

- 含有大量唯一值的列，如 ID 字段。
- 结果集很小的查询列。

7.【**解析**】 例如：用户对一张表中的许多记录都加了锁，这时系统会自动将它们升级为表级锁，原来加的所有的记录级锁都可以被释放，这样，锁的数目减少了，系统的性能也就获得了提高。至于何时升级，如何升级，完全取决于各数据库系统的具体实现。

采用锁升级，锁的粒度提高了，并发度也可能因此而降低，而且死锁的概率也大大增加了。假设两个事务都对同一张表中的许多记录加了锁，现在系统决定将其中一个事务的锁升级为表级锁，它将不得不等待另一个事务释放所有的锁，如果另一个事务也试图升级时，死锁就发生了。

8.【**解析**】 主机环境，如软硬件等；客户/服务器环境；互联网计算环境。

9.【**解析**】 模式（Schema），也称逻辑模式，是数据库中全体数据的逻辑结构和特征的描述。是所有用户的公共数据视图。

① 外模式，也称子模式或用户模式，是数据库用户能够看见和使用的局部数据的逻辑结构和特征的描述，是数据库用户的数据视图，是与某一应用有关的数据的逻辑表示。

② 内模式，也称存储模式，是数据物理结构和存储方式的描述，是数据在数据库内部的表示方式。

10.【**解析**】 功能强大，能准确地表示业务数据，容易使用和维护，对最终用户操作的响应时间合理，便于数据结构的改进，便于数据数据的检索和修改，，由于设计缺陷所造成的停机时间最少，很少的数据库维护工作，有效的安全机制，冗余数据少或没有，便于数据库的备份和恢复，数据库结构对最终用户透明。

11. 函数依赖为：

【**解析 1**】 订单号→（订货日期，客户名称，厂商名称，厂商地址，产品名称，数量，

价格，折扣）。

厂商名称→厂商地址。

【解析 2】 分解后的实体为：

订单（订单号，订货日期，客户名称，厂商名称，产品名称，数量，价格，折扣）。

厂商（厂商名称，厂商地址）。

12.【解析】

```
CREATE OR REPLACE VIEW V_CONTRACT_SALE AS
    SELECT * FROM CONTRACT WHERE SALE='销售人员姓名';
CREATE OR REPLACE VIEW V_CONTRACT_TECH AS
    SELECT NO，SALE，CUSTOMER，PDT_CONFIG，SERVICE FROM CONTRACT;
CREATE OR REPLACE VIEW V_CONTRACT_FINANCE AS
    SELECT NO， SALE， CUSTOMER， PRICE， PAY FROM CONTRACT;。
```

13.【解析 1】 满足需求的 E-R 图如图 1-22 所示。

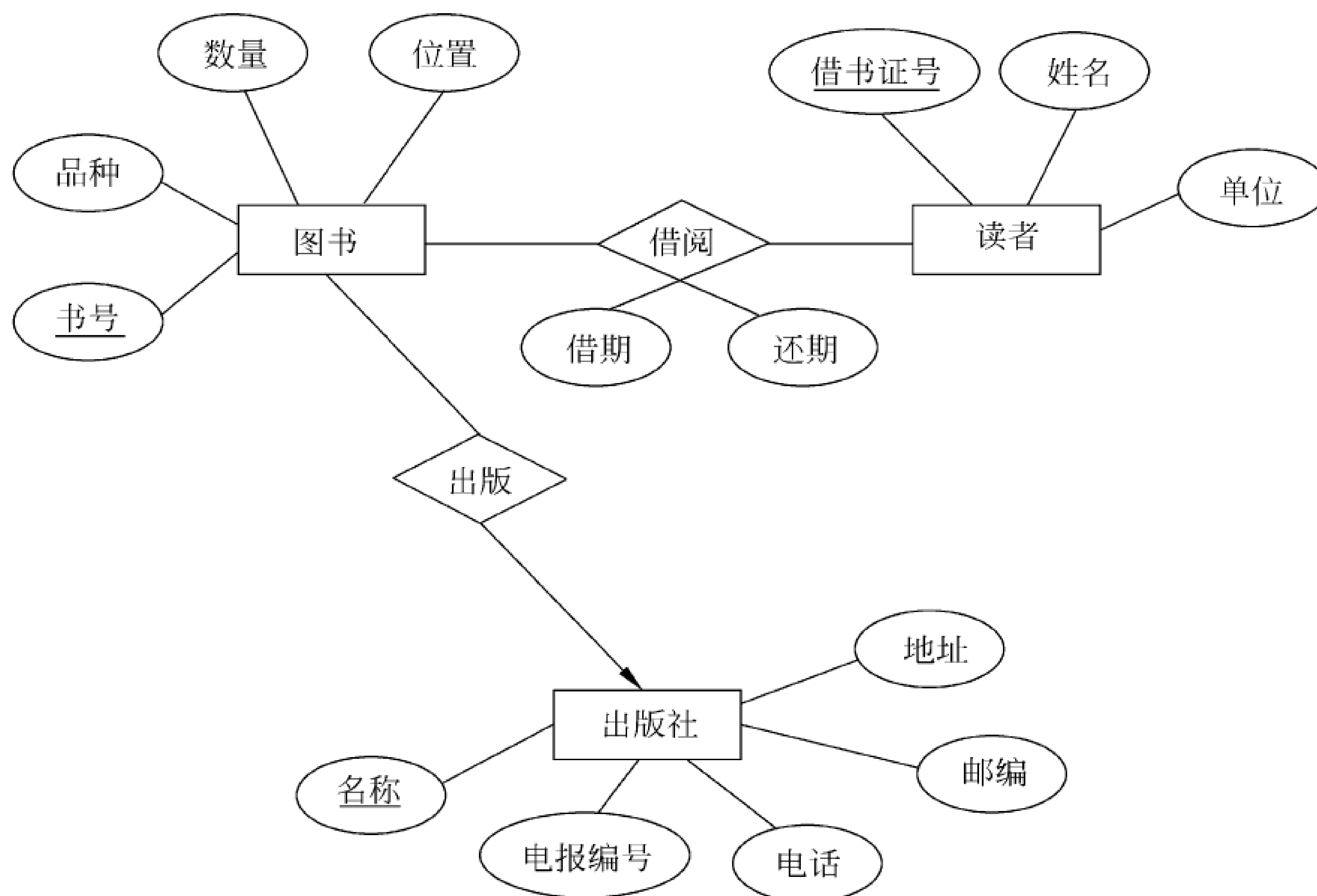


图 1-22 满足需求的 E-R 图

【解析 2】 等价的关系模式如下所示。

图书（<u>书号</u>，品种，数量，位置）

读者（<u>借书证号</u>，姓名，单位）

出版社（<u>名称</u>，电报编号，电话，邮编，地址）

借阅（<u>书号</u>，<u>借书证号</u>，借期，还期）

出版（书号，名称）

14.【解析】 A+=ABCD

B+=BCD

C+=C

D+=D

由此可知 A 是键码。

15.【解析】 封锁是指某事务在对某数据对象进行操作以前，先请求系统对其加锁，成功加锁之后该事务就对该数据对象有了控制权，只有该事务对其进行解锁之后，其他的事务才能更新它。

DBMS 提供的基本封锁类型有两种：

① 排它锁（X 锁）。某事务对某数据加 X 锁后，其他事务不能再加任何类型的锁，不能读和修改加了 X 锁的数据

② 共享锁（S 锁）。某事务对数据加了 S 锁，则该事务可以读取数据，其他事务可以在对加了 S 锁的数据加 S 锁，但不能加 X 锁，从而其他事务可以读取数据。

封锁数据对象的大小称为封锁的粒度。封锁粒度越大，系统的并发程度越低，系统开销越少；封锁粒度越小，系统的并发程度越高，系统开销越大。

16.【解析】 在关系模式中，函数依赖不能表示属性值之间的一对多联系，这些属性之间有些虽然没有直接关系，但存在间接的关系，把没有直接联系、但有间接的联系称为多值依赖的数据依赖。例如，教师和学生之间没有直接联系，但教师和学生可通过系名，或任课把教师和学生联系起来。

17.【解析】 把数据库应用系统从开始规划、系统分析、系统设计、实施、投入运行后的维护到最后由新的系统替换原有的数据库系统的整个期间。

18.【解析】

① 系统本身的需求是不断变化的；

② 由于用户缺少计算机信息系统设计方面的专业知识，要准确表达需求很困难；

③ 通过需求分析可以沟通用户与设计人员。

19.【解析】 基于对象关系数据模型的 DBS 称为对象关系数据库系统（ORDBS）。

20.【解析】 由于概念模式和内模式的两级的数据结构可能不一致，即记录类型、字段类型的命名、组成可能不一致，用这个映像说明概念记录和内部记录间的对应性。

21.【解析】 当对数据库的概念模式进行修改时，内模式尽可能保持不变或尽量少的作修改，即对概念模式的修改尽量不影响外模式和应用程序，称数据库达到了逻辑数据独立性。

22.【解析】 在多用户共享系统中，多个用户同时对同一数据进行操作称为并发操作。

23.【解析】 S#→SD 学生学号，决定其所在系名；

SD→MN 系名决定了其所在系的系主任；

S＃，CN→G 学号和课程决定该生此课程的成绩。

24.【解析】 数据库技术的一个特点是数据共享，但多个用户同时对同一个数据的并发操作可能会破坏数据库中的数据，数据库的并发控制能防止错误发生，正确处理好多用户、多任务环境下并发操作。

25.【解析】 物理数据独立性：指数据库物理结构，即数据的组织和存储、存取方法、外部存储设备等，发生改变时，不会影响到逻辑结构，而用户使用的是逻辑数据，所以不必改动程序。

逻辑数据独立性：数据库的逻辑结构发生改变时，用户也不须改动程序，就像数据库并没发生变化一样。

26.【解析】 最左列表示事务 T_1 已经获得的数据对象上的锁的类型，其中的“－”表示没有加锁；最上面一行表示另一事务 T_2 对同一数据对象发出的封锁请求。T_2 的封锁请求能否被满足用矩阵中的 Y 和 N 表示，其中 Y 表示事务 T_2 封锁请求与 T_1 已经获得的锁相容，封锁请求可以满足。N 表示事务 T_2 封锁请求与 T_1 已经获得的锁冲突，封锁请求被拒绝。如表 1-1 所示。

表 1-1　封锁类型的相容矩阵

T_1 \ T_2	X 锁	S 锁	－
X 锁	N	N	Y
S 锁	N	Y	Y
－	Y	Y	Y

27.【解析 1】 $\prod_{cnum}$（C）$-\prod_{cnum}$（$\sigma_{sname='李强'}$（S）∞SC）

【解析 2】 $\prod_{sname}$（S∞SC∞C$\sigma_{teacher='李文'}$）

28.【解析 1】 {$t^{(2)}$|（$\exists r^{(4)}$）（S（r）∧t[1]=r[1]∧t[2]=r[2]∧r[3]>21∧r[4]='女'）

【解析 2】 {$t^{(1)}$|（$\exists u^{(4)}$）（$\exists v^{(3)}$）（$\exists w^{(3)}$）

（S（u）∧SC（v）∧C（w）∧t[1]=u[2]∧u[1]=v[1]∧v[2]=w[1]∧w[2]='数据库系统概论'）}

29.【解析】 AE^+=AEDCI，因为 B 不属于 AE^+，所以 AE 不是键码。

30.【解析】 共享性封锁（共享锁，或称 S 锁），也称读锁（RLOCK），即若事务 T 对数据对象 A 加上 S 锁，则事务 T 可以读取 A 但不能修改 A，其他事务只能对 A 加 S 锁，而不能加 X 锁，直到 T 释放 A 上的 S 锁。这就保证了其他事务可以读 A，但在 T 释放 A 上的锁之前不能修改 A。

排他性封锁（排他锁，或称 X 锁），也称写锁（Wlock），即若事务 T 对数据对象 A 加上 X 锁，则只允许 T 读取和修改 A，其他任何事务都不能再对 A 加任何类型的锁，直到 T 释放 A 上的锁。这就保证了其他事务在 T 释放 A 上的锁之前不能在读取和修改 A。

31.【解析】 关系模型中有 3 种完整性约束：实体完整性、参照完整性和用户定义完

整性。

① 实体完整性规则：关系中的主键不能为空值（NULL）。

② 参照完整性规则：表的外键必须是另一个表主键的有效值，或者是空值。

④ 用户定义完整性规则：用户按照实际的数据库运行环境要求，对关系中的数据所定义的约束条件，它反映的是某一具体应用所涉及的数据必须要满足的条件。

32.**【解析】** 实体之间的联系类型比较复杂，一般分为一对一、一对多、多对多 3 类。

① 一对一联系（1∶1）

如果对于实体集 A 中的每个实体，实体集 B 中至多有一个（可以没有）与之相对应；反之亦然，则称实体集 A 与实体集 B 具有一对一联系，记作 1∶1。

② 一对多联系（1∶n）

如果对于实体集 A 中的每个实体，实体集 B 中有 n 个实体（$n \geq 0$）与之相对应；反过来，实体集 B 中的每个实体，实体集 A 中至多只有一个实体与之联系，则称实体集 A 与实体集 B 具有一对多联系，记作 1∶n。

③ 多对多联系（m∶n）

如果对于实体集 A 中的每个实体，实体集 B 中有 n 个实体（$n \geq 0$）与之相对应；反过来，实体集 B 中的每个实体，实体集 A 中也有 m 个实体（$m \geq 0$）与之联系，则称实体集 A 与实体集 B 具有多对多联系，记作 m∶n。

33.**【解析】** 首先，应该统一术语，也就是说，必须了解并使用用户的术语。

其次，不要试图一次就完全了解用户的需求，而要在系统开发过程中不断请用户参与，以一种螺旋的、渐进的方式获取用户需求。

其三，使用使用合适的需求工具。

当然，需求分析本身就是一个非常复杂的过程，还有很多注意的问题，考生应在平时工作中多总结。

第 2 章　数据库应用系统设计

本章提示

数据库应用系统设计是指数据库前端应用的设计。应用系统设计是软件开发过程中的一个重要阶段，在这个阶段将从计算机实现的逻辑角度开发针对用户需求的解决方案。这一解决方案是一个高级的抽象方案。高层设计要设计出各主要部分，并说明在技术上如何工作：相互间的协作；所需外在的硬件和软件环境；内在环境。设计的内容包括：应用系统结构的设计、输入输出的设计、物理数据设计、安全体系设计等。除了设计方面的内容，本章还涉及应用程序开发中的关键点，如模块划分、文档编写、以及设计和程序评审相关的注意点。

如图 2-1 所示是本章的知识框图。

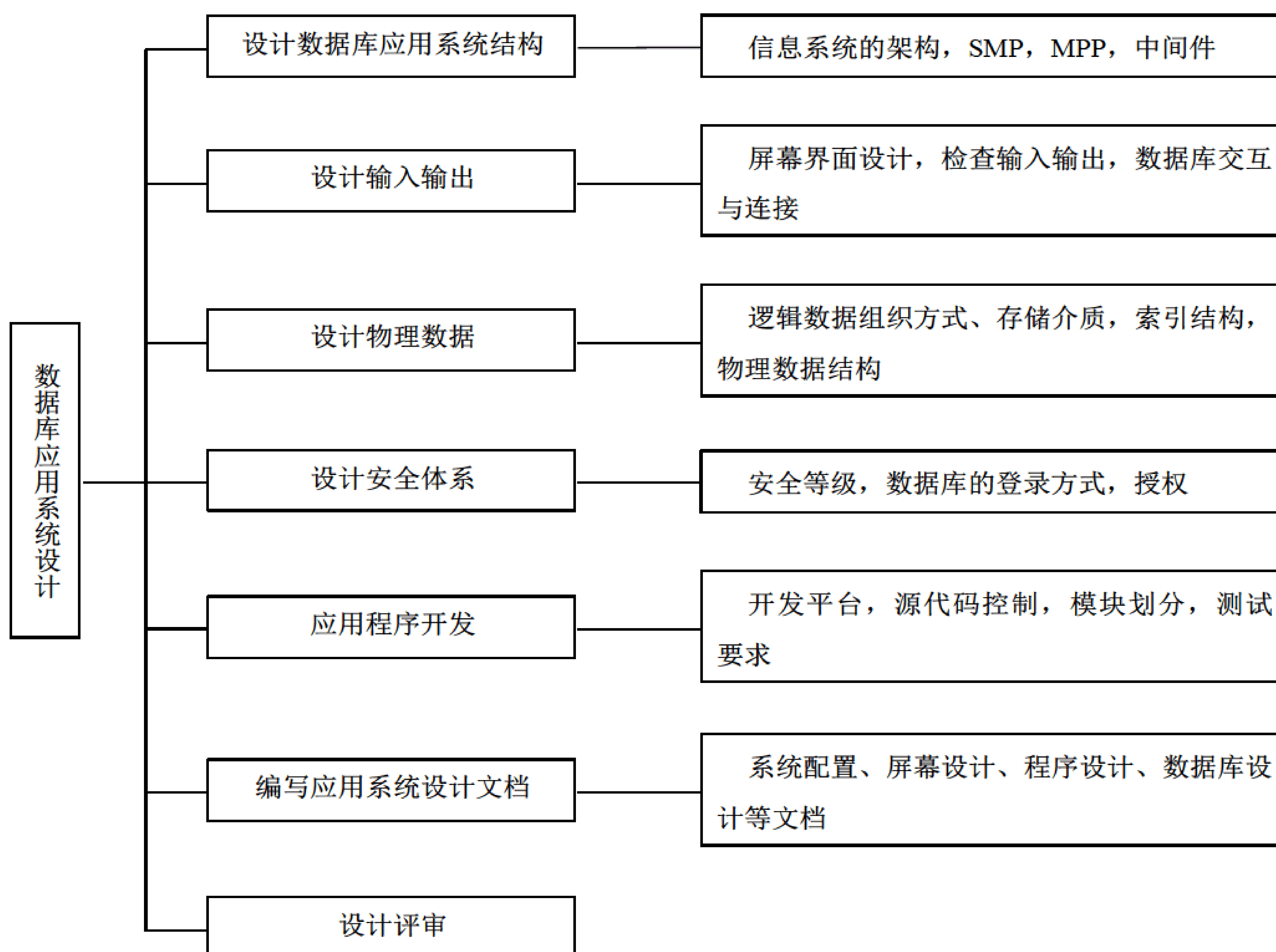


图 2-1　数据库应用系统设计知识框图

2.1　设计数据库应用系统结构

设计数据库应用系统结构时首先应该考虑信息系统的架构，是选择 B/S 结构还是选择 C/S 结构，DBMS 是选择桌面数据库还是选择远程数据库；在多用户数据库环境下是选择文件服务器体系结构，还是选择 Client/Server 体系结构；如果是多层的系统构件，是否选用适当的中间件等等，这些问题都是需要考虑的。当然，系统如果需要处理海量数据和大规模数据库时，硬件选型时并行计算机体系结构（SMP、MPP）也是需要重点考虑的。

如图 2-2 所示是本节的知识框图。

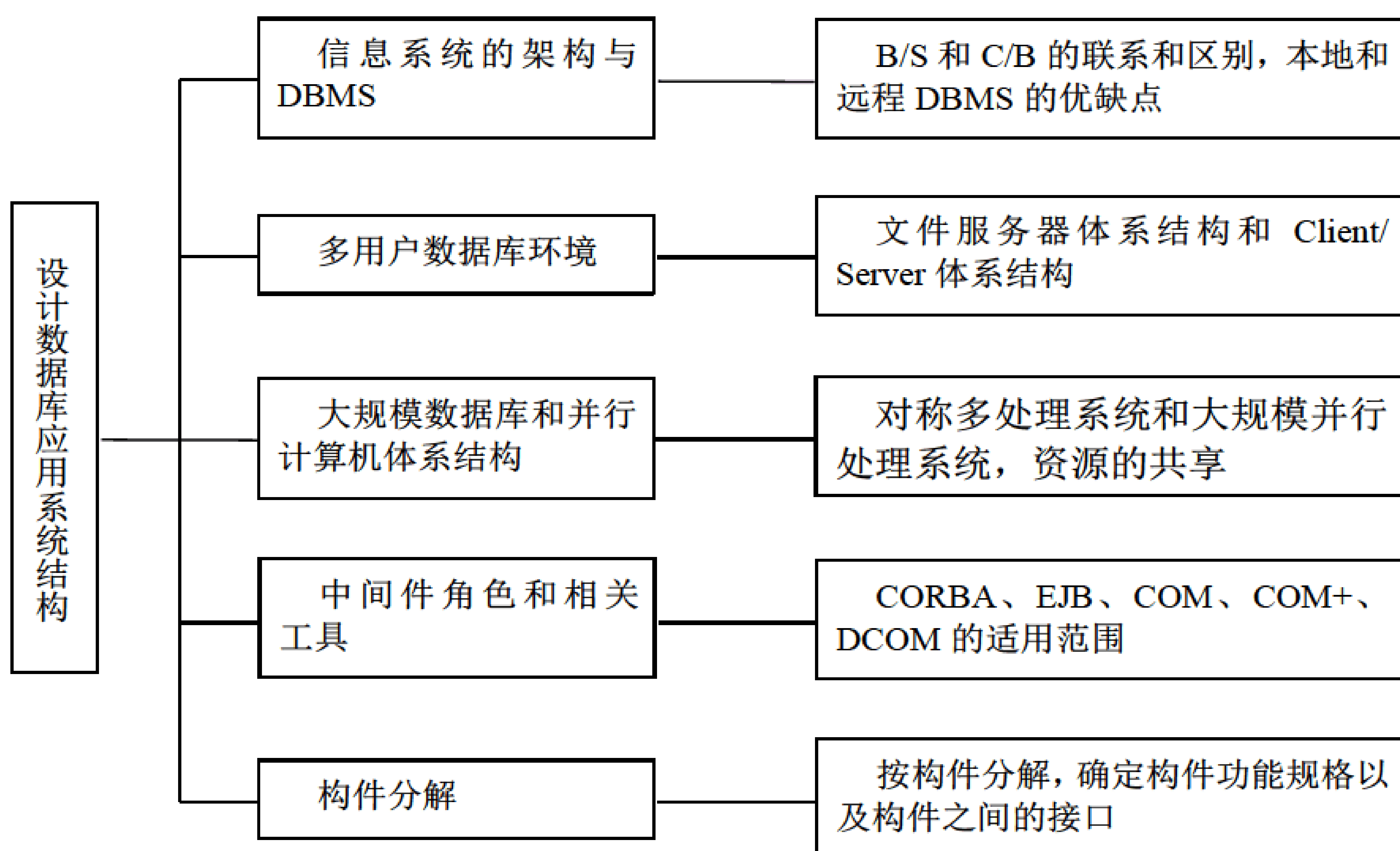


图 2-2　设计数据库应用系统结构知识框图

1．知识点提炼

（1）信息系统的架构（如 Client/Server）与 DBMS

信息系统的架构通常采用客户/服务器体系结构（Client/Server Architecture，C/S）和浏览器/服务器结构（Browser/Server Architecture，B/S）等。

最简单的 C/S 体系结构的数据库应用由两部分组成，即客户应用程序（也称为前台程序）和数据库服务器程序（也称为后台程序)。运行数据库服务器程序的计算机被称为应用服务器，运行数据库客户应用程序的计算机被称为客户机。一旦服务器程序被启动，就随时等待响应客户程序发来的请求。当需要对数据库中的数据进行任何操作时，客户程序自动地寻找服务器程序，并向其发出请求，服务器程序则根据预定的规则做出应答，返回结果。在典型的 C/S 数据库应用中，数据的存储管理功能是由服务器程序独立进行的，通常

情况下，把前台应用所不能违反的规则在服务器程序中集中加以约束。C/S 结构是建立在局域网基础上的，一般建立在专用的网络上，小范围内的网络环境，局域网之间再通过专门服务器提供链接和数据交换服务。

传统的管理信息系统一般采用 C/S 结构来完成。这一架构的缺点是；业务逻辑位于客户端，每完成一项事务，都要频繁地访问数据库，使得网络上数据流量非常大，对于慢速连接的用户，甚至无法使用。

为弥补 C/S 结构的缺陷，提出了三层或多层架构：客户机—中间件（应用服务器）—数据库服务器（Client—Middle Ware—Database Server）。在三层架构中，业务逻辑放置于中间件服务器上，大量的数据流位于中间件和数据库之间，而客户机只是简单地发出请求，中间件接受请求后进行事务处理并将处理的结果返回给客户机。

实际上，B/S 结构是三层架构的一种，所不同的是客户端就是目前几乎每台计算机中都有的网络浏览器，而中间件则是 Web 应用服务器，主要的业务逻辑均由位于 Web 应用服务器上的 Servlet 和 JSP 程序或 EJB 来处理。

B/S 结构是随着 Internet 技术的兴起，对 C/S 结构的一种变化或者改进的结构。B/S 结构，主要是利用不断成熟和发展的 WWW 浏览器技术，并结合浏览器的多种 Script 语言（VBScript、JavaScript 等）和 ActiveX 技术，是一种全新的信息系统构架技术。B/S 结构的优点在于简化了客户端程序的开发，利用通用的浏览器实现原来需要用复杂的专用客户端软件才能实现的强大功能，节约了开发成本。在 B/S 结构下，用户界面即客户端程序，完全通过 WWW 浏览器来实现，部分事务逻辑在客户端实现，但主要事务逻辑在服务器端实现。B/S 结构建立在广域网的基础上，采用星状拓扑结构建立企业内部通信网络或利用 Internet 虚拟专网（VPN）。

数据库管理系统（DBMS）是处理数据库访问的软件。用户使用某种数据库语言（如 SQL）发出访问请求，DBMS 接受请求后分析请求，检查用户外模式、相应的外模式/模式映像、概念模式、概念模式/内模式映像和存储结构定义，然后执行对数据库的必要操作。

DBMS 的功能包括数据定义、数据操纵、数据安全性和完整性、数据恢复和并发控制、优化和执行。

数据定义，即接受数据定义的源形式，并将它转换成相应的目标形式。

数据操纵，即提供对数据的检索、更新、插入、删除。数据操纵包括 DML 处理器/编译器、查询语言处理器（处理非计划查询）。

数据安全性和完整性，即监视用户请求，拒绝违反安全性和完整性约束的请求。

数据恢复和并发控制，即由事务管理器/事务处理监控器确保数据库面临故障时能够正确地恢复，并发访问正确地进行。

优化和执行，即有效处理非计划查询，由优化器完成。优化后的请求在运行管理器的控制下执行。

（2）多用户数据库环境（Client/Server 体系结构）

在多用户数据库环境中，通过对服务功能的分布实现分工服务。在 Client/Server 体系结构中，客户机负责管理用户界面，接收用户数据，处理应用逻辑，生成数据库服务请求，然后将用户的服务请求发送给服务器，并接收从服务器返回的结果，最后再将这些结果按一定的格式返回给用户。服务器接收来自客户机的服务请求，处理这些请求，然后将处理结果，包括执行状态以及数据库访问的结果数据等返回给客户机，同时，服务器还要进行数据库完整性和安全性检查，维护数据，支持并发访问控制等。

其中数据库服务器的主流产品有 Microsoft 公司的 SQL Server、SYBASE 公司的 Sybase、ORACLE 公司的 Oracle、INFORMIX 公司的 Informix 和 IBM 公司的 DB2。

（3）大规模数据库和并行计算机体系结构（SMP、MPP）

SMP（Symmetric Multi Processing）即对称多处理系统，内有许多紧耦合多处理器，该系统的特点是共享所有资源。

与 SMP 相对立的标准是 MPP（Massively Parallel Processing），即大规模并行处理系统，该系统是由许多松耦合处理单元组成的，每个处理单元内的 CPU 都有自己私有的资源，如总线、内存、硬盘等，在每个处理单元内都有操作系统和管理数据库的实例复本，该结构的特点是不共享资源。

（4）中间件角色和相关工具

为了尽可能地缩短信息系统的开发周期，集软件复用、分布式对象计算、企业级应用开发等技术为一体的“基于中间件的软件开发”（Component Based Sortware Development，CBSD）应运而生，该技术以软件架构为组装蓝图，以可复用软件构件为组装模块，支持组装式软件的复用，大大提高了软件生产效率和软件质量。

中间件技术是伴随网络而发展起来的一种面向对象的技术。中间件是位于操作系统和应用软件之间的通用服务，是独立于硬件或数据库厂商，处于硬件产品和数据库产品的中间，实现其互连的一类软件，是客户方与服务方之间的连接件，是需要进行二次开发的中间产品。中间件的主要作用是用来屏蔽网络硬件平台的差异性和操作系统与网络协议的异构性，使应用软件能够比较平滑地运行于不同平台上。同时，中间件在负载平衡、连接管理和调度方面起了很大的作用，使企业级应用的性能得到大幅提升，满足了关键业务的需求。

中间件作为存在于系统软件与应用之间的特殊层次，抽象了典型的应用模式，从而使应用软件制造者可以将思路更多地放在业务逻辑上，并基于标准的形式进行开发，使软件构架化成为可能。一些标准的推出，进一步使中间件成为可复用构件的运行框架，加速了软件复用的进程。

当前主流的分布计算技术平台，主要有 OMG 的 CORBA、Sun 的 J2EE 和 Microsoft DNA 2000。它们都是支持服务器端中间件技术开发的平台，分别阐述如下。

CORBA 分布计算技术是由绝大多数分布计算平台厂商所支持和遵循的系统规范技术，具有模型完整、先进，独立于系统平台和开发语言，被支持程度广泛的特点，已逐渐

成为分布计算技术的标准。CORBA 的特点是大而全，互操作性和开放性非常好，其缺点是庞大而复杂，技术和标准的更新相对较慢。

COBRA 标准主要分为 3 个层次：对象请求代理、公共对象服务和公共设施。最底层是对象请求代理 ORB，规定了分布对象的定义（接口）和语言映射，实现对象间的通信和互操作；在 ORB 之上定义了很多公共对象服务，可以提供诸如并发服务、名字服务、事务（交易）服务、安全服务等服务；最上层的公共设施则定义了组件框架，提供可直接被业务对象使用的服务，规定业务对象有效协作所需的协定规则。

CORBA 规范增加了面向 Internet 的特性、服务质量控制和 CORBA 构件模型（CORBA Component Model）。

Internet 集成特性包括针对 IIOP 传输的防火墙和可内部操作的定义了 URL 命名格式的命名服务。

服务质量控制包括具有质量控制的异步消息服务，一组针对嵌入系统的 CORBA 定义，一组关于实时 CORBA 与容错 CORBA 的请求方案。

CORBA 构件模型技术是在支持 POA 的 CORBA 规范基础上，结合 EJB 当前规范的基础上发展起来的。CORBA 构件模型是 OMG 组织制定的一个用于开发和配置分布式应用的服务器端中间件模型规范。

为了推动基于 Java 的服务器端应用开发，Sun 公司于 1999 年底推出了 Java2 技术及相关的 J2EE 规范。J2EE 的目标是提供与平台无关的、可移植的、支持并发访问和安全的、完全基于 Java 的开发服务器端中间件的标准。

J2EE 给出了完整的基于 Java 语言开发面向企业分布应用规范，在分布式互操作协议上，J2EE 同时支持 RMI 和 IIOP，而在服务器端分布式应用的构造形式，则包括 Java Servlet、JSP（Java Server Page）、EJB 等多种形式，以支持不同的业务需求。

EJB 是 Sun 推出的基于 Java 的服务器端构件规范 J2EE 的一部分，推出之后得到了广泛的发展，已经成为应用服务器端的标准技术。从企业应用多层结构的角度，EJB 是业务逻辑层的中间件技术。EJB 技术是在 Java Bean 本地构件基础上发展的面向服务器端分布应用构件技术，它基于 Java 语言，提供了基于 Java 二进制字节代码的重用方式。EJB 给出了系统的服务器端分布构件规范，包括构件、构件容器的接口规范、构件打包、构件配置等的标准规范内容。

Microsoft DNA 2000（Distributed interNet Applications）是 Microsoft 公司在推出 Windows 2000 系列操作系统平台的基础上，在扩展了分布计算模型以及改造 Back Office 系列服务器端分布计算产品后发布的分布计算体系结构和规范。

在服务器端，DNA 2000 提供了 ASP、COM、Cluster 等的应用支持。DNA 2000 融合当今最先进的分布计算理论和思想，如事务处理、可伸缩性、异步消息队列、集群等内容。DNA 使得开发可以基于 Microsoft 平台的服务器构件应用，其中如数据库事务服务、异步通信服务和安全服务等，都由底层的分布对象系统提供。

DNA 2000 是单一厂家提供的分布对象构件模型，其优点是开发者可以使用同一厂家提供的系列开发工具，其缺点是依赖于 Microsoft 的操作系统平台，所以在其他开发系统平台比如 UNIX、Linux 无法使用。

（5）构件分解

软件构件是指应用系统中可以明确辨识的构成成分，包括源代码、需求、系统和软件的需求规约、系统和软件的构架、文档、测试计划、测试案例和数据以及其他开发活动有用的信息。基于构件的开发来自于利用构件生产应用软件的概念。开发者在设计和详细描述阶段，使用内部开发的构件和公开的构件来为要开发的应用软件提供尽可能多的功能，然后编写其他的构件来连接代码，把构件一一连接起来，使它们成为一个整体，实现系统功能。可以把新编写的构件放进知识库中，其他人就可以直接使用这些构件的功能。这种做法有效提高了软件重用的效率，降低了软件的开发成本。

构件的基本属性如下所述。

① 构件是可独立配置的单元，构件必须自包容。

② 构件强调与环境和其他构件的分离，构件的实现是严格封装的，外界没机会或没必要知道构件内部的实现细节。

③ 构件可以在适当的环境中被复合使用，构件需要提供清楚的接口规范，可以与环境交互。

④ 构件不应当是持续的，即构件没有个体特有的属性，理解为构件不应当与自身副本区别，在任何环境中，最多仅有特定构件的一份副本。

构件技术的关键点涉及构件做什么、构件交互的规则、构件存在的环境等，相应在现实中有各种支撑性的技术，分别列举如下。

① 构件模型，研究构件的本质特征及构件间的关系。

② 构件描述语言，以构件模型为基础，解决构件的精确描述、理解和组装问题。

③ 构件分类与检索，研究构件的分类策略、组织模式及检索策略，建立构件库系统，支持构件的有效管理。

④ 构件复合组装，包括源代码级的组装和基于构件对象互操作性的运行级组装。

⑤ 标准化，包括构件模型的标准化和构件库的标准化。

⑥ 软件构架，研究如何快速、可靠地应用可复用构件系统进行系统构造的方式，着重于软件系统自身的整体结构和构件间的互连。

构件是一种前沿的软件设计思想，对整个软件行业的发展有着至关重要的推动作用。而中间件作为应用软件系统集成的关键技术，保证了构件化思想的实施，并为构件提供了真正的运行空间。中间件领域工业化标准的制定、统一及实现，使基于构件的应用开发成为可能。反过来，构件对新一代中间件产品也起到促进作用。

中间件是构件存在的基础。构件技术在最初时更多是作为一种思想存在，进而才在一些关键的环节上发展出解决问题的技术分支。构件的存在某种程度上极大地依赖了构架技

术、环境、基础设施、计算平台，只有在适当的构架中，软件才有可能被抽象和隔离，最终成为构件。因此，单独讨论构件是抽象而空洞的。构架不是操作系统、数据库或网络协议，也不完全是应用，而是在某种特定意义上的构件运行容器，层次上介于应用和基础设施之间。中间件，从本质上是对分布式应用的抽象，因而抛开了与应用相关的业务逻辑的细节，保留了典型的分布交互模式的关键特征。经过抽象，将纷繁复杂的分布式系统经过提炼和必要的隔离后，以统一的层面形式呈现给应用。应用在中间件提供的环境中可以更好地集中于业务逻辑上，并以构件化的形式存在，最终自然而然地在异构环境中实现良好的协同工作。

构件要求有很好的业务自包容性，应用开发者可以按照不同的业务进行功能的划分，体现为不同的接口或交互模式。构架的目标是提供业务的分隔和包容性。

构件对外发生作用或构件间的交互，都是通过规范定义的接口进行，构件使用者只需要知道构件的接口，而不关心其内部实现，这是设计与实现分开的关键。构架就应当提供构件交互的规则，并基于这些规则实现类似容器的标准环境。

2．难点分析

C/S、B/S 的选型是本章节的一个难点，下面详细讲述一下它们的区别和优缺点。C/S 是建立在局域网的基础上的，B/S 一般是建立在广域网的基础上的。

① 硬件环境不同。C/S 一般建立在专用的网络上，小范围的网络环境，局域网之间通过专门服务器提供链接和数据交换服务。B/S 建立在广域网之上的，不必是专门的网络硬件环境，例与电话上网，租用设备；信息自己管理；有比 C/S 更强的适应范围，一般只要有操作系统和浏览器就可以。

② 对安全要求不同。C/S 一般面向相对固定的用户群，对信息安全的控制能力很强。一般高度机密的信息系统采用 C/S 结构适宜。可以通过 B/S 发布部分可公开信息；B/S 建立在广域网之上，对安全的控制能力相对弱，面向是不可知的用户群。

③ 对程序架构不同。C/S 程序可以更加注重流程，可以对权限多层次校验，对系统运行速度较少考虑。B/S 对安全以及访问速度的多重的考虑，建立在需要更加优化的基础之上；比 C/S 有更高的要求 B/S 结构的程序架构是发展的趋势，从 MS 的.Net 系列的 BizTalk 2000、Exchange 2000 等，全面支持网络的构件搭建的系统；Sun 和 IBM 推出的 JavaBean 构件技术等，使 B/S 更加成熟。

④ 软件重用不同。C/S 程序可以不可避免地整体性考虑，构件的重用性不如在 B/S 要求下的构件的重用性好。B/S 对的多重结构，要求构件相对独立的功能；能够相对较好地重用。

⑤ 系统维护不同。系统维护是软件生存周期的一部分，需要考虑开销的大小。C/S 程序由于整体性，必须整体考察，处理出现的问题以及系统升级。B/S 系统由构件组成，方面构件可以个别更换，实现系统的无缝升级，将系统维护开销减到最小；用户从网上自己下载安装就可以实现升级。

⑥ 处理问题不同。C/S 程序可以处理用户面固定，并且在相同区域，安全要求高需求，与操作系统相关的，应该都是相同的系统。B/S 建立在广域网上，面向不同的用户群，分散地域，这是 C/S 无法做到的，与操作系统平台关系最小。

⑦ 用户接口不同。C/S 多是建立的 Window 平台上，表现方法有限，对程序员普遍要求较高。B/S 建立在浏览器上，有更加丰富和生动的表现方式与用户交流，并且大部分难度减低，减低开发成本。

3．典型例题

【例题 2-1】 阅读以下有关 MTS 的叙述，回答问题。

MTS（Microsoft Transaction Server）是微软为其 Windows NT 操作系统推出的一个中间件产品，由于它具有强大的分布事务支持、安全管理、资源管理和多线程并发控制等特性，使其成为在 Windows 平台上开发大型数据库应用系统的首选产品。

某单位一信息项目的数据库平台采用了 SQL Server 2000 作为后端平台，前端选择了 Microsoft 公司的 Visual Studio 作为开发工具，采用基于 MTS 的三层体系结构。由于 MTS 屏蔽了底层实现的复杂性，极大地简化了这类应用的开发，程序员可以将精力集中在业务逻辑上，因而有效地提高了软件的开发效率。

【问题】 根据自己的项目经验，列举基于 MTS 开发多层数据库应用系统的步骤（100 字以内）。

【解析】 基于 MTS 开发多层数据库应用系统的步骤如下所示。

第一步，开发 MTS 组件提供服务，程序员可以用任何一种支持 COM 的语言编写 MTS 组件，如 VB、VC 等。

第二步，分发 MTS 组件到 MTS 软件包中，并且把 MTS 软件包安装到 MTS 环境之中。

第三步，编写客户端程序调用执行在 MTS 环境之中的 MTS 组件，以取得服务。

【例题 2-2】 阅读以下有关中间件的叙述，回答问题。

在某数据库项目开发初期，项目经理召集所有技术人员开会，就中间件应用提出以下一些看法。

① 中间件是介于应用系统和系统软件之间的一类软件，它使用系统软件所提供的基础服务（功能），衔接网络上应用系统的各个部分或不同的应用，能够达到资源共享、功能共享的目的。

② 中间件屏蔽了底层操作系统的复杂性，使程序开发人员面对一个简单而统一的开发环境，减少程序设计的复杂性，将注意力集中在自己的业务上，不必再为程序在不同系统软件上的移植而重复工作，从而大大减少了技术上的负担。

③ 中间件作为新层次的基础软件，其重要作用是将不同时期、在不同操作系统上开发应用软件集成起来，彼此像一个天衣无缝的整体协调工作，这是操作系统、数据库管理系统本身做不了的。中间件的这一作用，使得在技术不断发展之后，以往在应用软件上的劳动成果仍然物有所用，节约了大量的人力、财力投入。

④ 在多数应用系统中，业务逻辑程序和应用逻辑程序只占少量的比例，而基础程序却占了大量的比例。如果以新一代的中间件系列产品来组合应用，同时配合以可复用的商务对象构件，则应用开发费用可节省。

⑤ 在使用中间件的应用系统，其初期投入的资金比同规模没使用中间件的应用系统多一些时间限制是所有应用系统开发项目的天敌，而基础软件的开发又是一件极耗时的工作，若使用标准商业中间件则可缩短开发周期。

⑥ 项目中途夭折、费用远远超过预算、无法准时完成项目和偏离既定的目标都是项目失败的标志。没有使用标准商业中间件的关键应用系统开发项目的失败率较高。

⑦ 借助标准的商业中间件，企业可以很容易地在现有或遗留系统之上或之外增加新的功能模块，并将它们与原有系统无缝集合。

⑧ 在使用中间件的应用系统中，因为购买中间件平台需要一部分资金，初期投入的资金比同规模的没使用中间件的应用系统多一些。

⑨ 依靠标准的中间件可以将现有的应用、新的应用和购买的商务构件融合在一起。

⑩ 由于系统同时对中间件的维护，在使用中间件的应用系统的后期维护量更大一些。

【问题】 在上述 10 条叙述中有两条是不正确的或不恰当的，请指出其序号，并各在 50 字以内简要说明理由。

【解析】 第 8 条，在使用中间件的应用系统中，虽然购买中间件平台需要一部分资金，但由于其他方面的投入相对减少，初期投入的资金比同规模的没使用中间件的应用系统少一些。

第 10 条，中间件平台本身是比较成熟的，不存在对中间件的维护。

【例题 2-3】 阅读以下有关 Web 应用开发的叙述，回答问题。

由于 Web 应用开发的独特性，应用开发平台成为众多厂商关注的焦点。目前市场上存在很多的 Web 应用标准、集成开发环境。流行的主要是 ASP、PHP、JSP 3 种。某企业的信息部门提出以下一些看法。

① ASP（Activex Server Pages）是由微软创建的 Web 应用开发标准，ASP 服务器已经包含在 IIS 服务器中，ASP 服务器将 Web 请求转入解释器中，在解释器中将所有 ASP 中的脚本进行分析，然后执行。

② ASP 本身可以编写 COM 对象以完成更多的功能，ASP 中的脚本是 VBScript。

③ ASP 的优点是安装配置方便，开发简单易学；开发工具功能强大。不足之处是 ASP 使用了组件因而将导致大量的安全问题；无法实现跨平台，只能应用于 Windows NT/2000。

④ PHP 由于其良好的性能及免费的特点，是目前互联网中应用非常流行的一种应用开发平台。

⑤ PHP 的优点是简单易学、跨平台、有良好数据库交换能力的开发语言；与 Apache 及其扩展库紧密结合；良好的安全性。

⑥ PHP 不足之处是安装配置复杂；缺少企业级的支持；作为自由软件，缺乏正规的

商业支持；无法实现商品化的商业开发。

⑦ JSP 的优点是可移植性好，支持多种平台；强大的可伸缩性；多样化与强大的工具支持。不足之处是安装配置管理较为复杂；运行速度较慢，建议开发中小型应用系统采用 JSP。

【问题】 在上述 7 条叙述中有两条是不正确的或不恰当的，请指出其序号，并各在 50 字以内简要说明理由。

【解析】 以下的两条是不正确的或不恰当的。

第 2 条：ASP 可以调用 COM 对象以完成更多的功能，但是 ASP 本身不能编写 COM；ASP 中的脚本是 VBScript 和 JScript。

第 7 条：JSP 优点是可移植性好，支持多种平台；强大的可伸缩性；多样化与强大的工具支持。不足之处是安装配置管理较为复杂；但是认为“运行速度较慢”的提法欠妥；一般建议开发大型应用系统采用 JSP。

【例题 2-4】 阅读以下有关 B/S 和 C/S 的叙述，回答问题 1 和问题 2。

MIS 在我国已有 20 多年的发展历程， 但真正普及应用还是近十年来的事。由于负责企业庞大而复杂的数据信息的管理，在企业生产经营管理中发挥了巨大的作用。MIS 的构架又有 B/S 和 C/S 两种。某企业的信息部门提出以下一些看法。

① 客户/服务器应用模式的特点是大都基于“胖客户机”结构下的两层结构应用软件。客户端软件一般由应用程序及相应的数据库连接程序组成。服务器端软件一般是某种数据库系统。

② 当前的实际应用中多数服务器就是一台数据库服务器，而客户端就是用快速开发工具编写的客户软件，通过 ODBC 或 ADO 同数据库服务器通信，组成一个应用系统。

③ B/S 结构的优点是客户机统一采用浏览器，这不仅让用户使用方便，而且使得客户机不存在安装维护的问题。当然软件发布和维护的工作不是自动消失了，而是转移到了 Web 服务器端。在 Web 服务器端，程序员使用脚本语言编写响应页面。

④ 在某些实际应用开发中，一般把简单查询和数据操作放在前端开发语言中，复杂的业务逻辑程序放在了数据库的存储过程中，这种方式也属于 C/S 结构，不过业内人员有时称这种方式为两层半结构。

⑤ C/S 应用模式的缺点是系统客户方软件安装维护困难、数据库系统无法满足成百上千的终端同时联机的需求、由于客户/服务器间的大量数据通信不适合远程连接，使其只能适合于局域网应用。

⑥ B/S 结构当前主要的浏览器是 Netscape Navigator 和 Internet Explorer，Netscape Navigator 和 Windows 捆绑销售，而 Internet Explorer 是可以免费下载的。国内大部分客户机基于 Internet Explorer，而服务器上的脚本使用 ASP、JSP 或 PHP 编写。

⑦ 采用 C/S 结构，服务器的操作系统不但可以为 Windows 系统服务器，也可以是 UNIX、Linux 服务器，除了服务器端安装及维护方式不同外，客户端安装及连接服务器方

式同连接 Windows 服务器方式没有区别。

⑧ B/S 结构客户机可以是 Windows（浏览器为 Internet Explorer）、Linux（浏览器为 Netscape Navigator）、UNIX（浏览器为 Netscape Navigator），而服务器可以是 Windows 服务器（Web 服务为 IIS5，数据库为 MySQL，脚本语言为 PHP）、也可以是 UNIX（Web 服务为 Apache，数据库 SQL Server 或 Access，脚本语言为 ASP）。

⑨ B/S 结构客户机同 Web 服务器之间的通信采用 HTTP 协议，由于 HTTP 协议是一种无连接的协议，浏览器只有在接受到请求后才和 Web 服务器进行连接，Web 服务器马上与数据库通信并取得结果，Web 服务器再把数据库返回的结果转发给浏览器，浏览器接收到返回信息后马上断开连接。由于真正的连接时间很短，这样 Web 服务器可以共享系统资源，为更多用户提供服务，达到可以支持几千、几万甚至于更多用户的能力。

⑩ C/S 结构一般应用于客户机在 50 台以下的 ERP 系统，客户机采用 Microsoft Visual Basic 或 Delphi 编写，服务器采用 SQL Server、DB2、Oracel 等大型数据库。

B/S 结构一般用于电子商务网站、大型公司企业网、客户机是无盘工作站的多客户机的系统。由于 HTML 语言的可扩展性好，有较好的打印和界面控制功能，所以主要用于网站建设。

【问题 1】 在上述 11 条叙述中有 3 条是不正确的或不恰当的，请指出其序号，并各在 50 字以内简要说明理由。

【解析】 3 条不正确的或不恰当的叙述如下所示。

第 6 条，B/S 结构当前主要的浏览器是 Netscape Navigator 和 Internet Explorer，Internet Explorer 和 Windows 捆绑销售，而 Netscape Navigator 是可以免费下载的。国内大部分客户机基于 Internet Explorer，而服务器上的脚本使用 ASP、JSP 或 PHP 编写。

第 8 条，而服务器可以是 Windows 服务器（Web 服务为 IIS5，数据库 SQL Server 或 Access，脚本语言为 ASP）、也可以是 UNIX（Web 服务为 Apache ，数据库为 MySQL，脚本语言为 PHP）。

第 11 条，由于当前 HTML 语言的局限性，其打印和界面控制不是很理想，所以主要用于网站建设。

【问题 2】 应用服务器分为基于中间件的应用服务器、基于 Web 的应用服务器和基于 DCOM/COM 的应用服务器。列举相应的产品。

【解析】 基于中间件的应用服务器代表为 IBM 的 CICS 和 BEA 的 Tuxedo；基于 Web 的应用服务器，代表为 IBM 的 WebSphere 和 BEA 的 Weblogic；基于 DCOM/COM 的应用服务器。代表为微软的 MTS 和 Borland 的 Midas。

【例题 2-5】 阅读以下有关 CORBA 的叙述，回答问题 1 和问题 2。

通用对象代理体系结构 CORBA（Common Object Request Broker Architecture）是对象管理组织所定义的用来实现现今大量硬件、软件之间互操作的解决方案，CORBA 也是迈向面向对象标准化和互操作的重要一步。CORBA 允许应用之间相互通信，而不管它们存

在于哪里以及是谁设计的。CORBA 1.1 于 1991 年由 OMG 发布，其中定义了接口定义语言（IDL）以及在对象请求代理（ORB）中实现客户对象与服务器对象之间交互的应用编程接口（API）。CORBA 2.0 于 1994 年发布，规定了各个供应商之间的 ORB 的通信规则。

【问题 1】 CORBA 标准主要分为 3 个部分，列举这 3 个部分的名称。

【解析】 接口定义语言（IDL）、对象请求代理（ORB）以及 ORB 之间的互操作协议 IIOP。

【问题 2】 CORBA 为今天的计算环境带来了真正的互操作性，使用 CORBA，用户可以透明地访问信息，并不需要知道信息存在于什么软件中、使用什么硬件平台，以及位于企业网络的什么地方，这种透明性的原理是什么？

【解析】 ORB 是对象之间建立 Client/Server 关系的中间件。使用 ORB，客户可以透明地调用一个服务对象上的方法，这个服务对象可以在本地，也可以在通过网络连接的其他计算机上。ORB 截获这一调用同时负责查找实现服务的对象并向其传递参数、调用方法、返回最终结果。客户并不知道服务对象位于什么地方，它的编程语言和操作系统是什么，也不知道不属于对象接口的其他系统部分。这样，ORB 在异构分布环境下为不同计算机上的应用提供了互操作性，并无缝地集成了多种对象系统。

在开发传统的 Client/Server 应用时，开发者使用他们自己设计的或一个公认的标准来定义用于设备之间通信的协议。协议的定义依赖于实现语言、网络传输和许多其他因素，而 ORB 的出现简化了这一过程。使用 ORB 时，协议是使用接口定义语言（IDL）定义的，而 IDL 是独立于语言的。并且 ORB 提供很强的灵活性，它使程序员选择最适合的操作系统、执行环境，甚至系统各个组件也可以采用不同的编程语言实现。更重要的是它允许现有组件的集成。在一个基于 ORB 的解决方案中，开发者可以使用与创建新对象一样的 IDL 对遗留系统进行建模，他们创建“包装”代码以在标准化的软件总线与遗留系统接口之间传递信息。

【例题 2-6】 阅读以下有关 OLTP 和 OLAP 的叙述，回答问题 1 到问题 3。

随着数据库技术的广泛应用，企业信息系统产生了大量的数据，如何从这些海量数据中提取对企业决策分析有用的信息成为企业决策管理人员所面临的重要难题。传统的企业数据库系统（管理信息系统）即联机事务处理系统（On-Line Transaction Processing，OLTP）作为数据管理手段，主要用于事务处理，但它对分析处理的支持一直不能令人满意。因此，人们逐渐尝试对 OLTP 数据库中的数据进行再加工，形成一个综合的、面向分析的、能更好支持决策制定的决策支持系统（Decision Support System，DSS）。联机分析处理的概念最早由关系数据库之父 E.F.Codd 于 1993 年提出。Codd 认为联机事务处理（OLTP）已不能满足终端用户对数据库查询分析的要求，SQL 对大型数据库的简单查询也不能满足用户分析的需求。用户的决策分析需要对关系数据库进行大量计算才能得到结果，而查询的结果并不能满足决策者提出的需求。因此，Codd 提出了多维数据库和多维分析的概念，即 OLAP。OLAP 委员会对联机分析处理的定义为：使分析人员、管理人员或执行人员能够从

多种角度对从原始数据中转化出来的、能够真正为用户所理解的、并真实反映企业维特性的信息进行快速、一致、交互地存取，从而获得对数据的更深入了解的一类软件技术。OLAP 的目标是满足决策支持或多维环境特定的查询和报表需求，它的技术核心是“维”这个概念，因此 OLAP 也可以说是多维数据分析工具的集合。

【问题 1】 OLAP 最终的数据来源与 OLTP 一样，均来自底层的数据库系统，但两者面对的用户群不同，数据内容的特点也不同。根据自己的项目经验，谈谈两者的区别主要有哪些？

【解析】 两者的区别主要如表 2-1 所示。

表 2-1　OLAP 和 OLTP 的主要区别

OLTP 数据	OLAP 数据
原始数据	导出数据
细节性数据	综合性和提炼性数据
当前值数据	历史数据
可更新	不可更新，但周期性刷新
一次处理的数据量小	一次处理的数据量大
面向应用，事务驱动	面向分析，分析驱动
面向操作人员，支持日常操作	面向决策人员，支持管理需要

【问题 2】 根据自己的项目经验，谈谈 OLAP 的特点和评价准则有哪些？

【解析】 OLAP 的特点可以用 5 个关键字来代表：Fast Analysis of Shared Multidimentional Information（FASMI）即共享多维信息的快速分析。这也是设计人员或管理人员用来判断一个 OLAP 设计是否成功的准则。

① Fast：系统响应用户的时间要相当快捷，要达到这个目标，数据库的模式应该朝着更广泛的技术发展，包括特殊的数据存储格式、预先计算和硬件配置等。

② Analysis：系统应能处理与应用有关的任何逻辑分析和统计分析，用户无须编程就可以定义新的专门计算，将其作为分析的一部分，并以用户理想的方式给出报告。用户可以在 OLAP 平台上进行数据分析，也可以连接到其他外部分析工具上，同时应提供灵活开放的报表处理功能，以保存分析结果。

③ Shared：这意味着系统要能够符合数据保密的安全要求，即使多个用户同时使用，也能够根据用户所属的安全级别，让他们只能看到他们应该看到的信息。

④ Multidimensional：OLAP 的显著特征就是它能提供数据的多维视图，提供对数据分析的多维视图和分析，包括对层次维和多重层次维的完全支持。

⑤ Information：不论数据量有多大，也不管数据存储在何处，OLAP 系统应能及时获得信息，并且管理大容量信息。这里有许多因素需要考虑，如数据的可复制性、可利用的磁盘空间、OLAP 产品的性能及与数据仓库的结合度等。

【问题 3】 OLAP 的基本多维分析操作有钻取（Drill-up 和 Drill-down）、切片（Slice）

和切块（Dice）、旋转（Pivot）等，请解释其具体含义。

【解析】 其具体含义如下。

① 钻取：是改变维的层次，变换分析的粒度。它包括向下钻取（Drill-down）和向上钻取（Drill-up）/上卷（Roll-up）。Drill-up 是在某一维上将低层次的细节数据概括到高层次的汇总数据，或者减少维数；而 Drill-down 则相反，它从汇总数据深入到细节数据进行观察或增加新维。

② 切片和切块：是在一部分维上选定值后，关心度量数据在剩余维上的分布。如果剩余的维只有两个，则是切片；如果有 3 个或以上，则是切块。

③ 旋转：是变换维的方向，即在表格中重新安排维的放置（例如行列互换）。

【例题 2-7】 阅读以下有关的叙述，回答问题 1 和问题 2。

公用对象请求代理（调度）程序体系结构（Common Object Request Broker Architecture，CORBA）是对象管理组织（Object Management Group）对应当今快速增长的软硬件的协同工作能力的要求而提出的方案。CORBA 允许应用程序和其他的应用程序通信，而不论它们在什么地方或者由谁来设计。

【问题 1】 在传统的客户/服务器程序中，开发者使用他们自己设计的或者公认的标准定义设备之间的协议。协议的定义依赖于实现的语言，网络的传输和其他许多因素。ORB 将这个过程简单化。CORBA 1.1 由对象管理组织在 1991 年发布，它定义了接口定义语言（IDL）和应用编程接口（API），从而通过实现对象请求代理（ORB）来激活客户/服务器的交互。CORBA 2.0 于 1994 年的 12 月发布，它定义 ORB 是一个中间件，它在对象间建立了客户/服务器的关系。请叙述 CORBA 对应用透明的基本原理。

【解析】 通过 ORB，一个客户可以很简单地使用服务器对象的方法而不论服务器是在同一计算机上还是通过一个网络访问。ORB 截获调用然后负责找到一个对象实现这个请求，传递参数和方法，最后返回结果。客户不用知道对象在哪里，是什么语言实现的，其操作系统以及其他和对象接口无关的东西。使用 ORB，协议的定义是通过应用接口，而该接口是接口定义语言（IDL）的一个实现，它和使用的编程语言无关。并且 ORB 提供了很大的灵活性，它让程序员选择最适当的操作系统，运行环境和设计语言来建设系统中每个组件；允许集成已经存在的组件 CORBA 是在面向对象标准化和互操作性道路上的一个信号。通过 CORBA，用户不必知道软硬件的平台和他们处在企业网的什么地方就可以操作。

【问题 2】 目前，CORBA 技术在银行、电信、保险、电力和电子商务领域都有广泛的应用。软件市场中能够见到的 CORBA 中间件产品很多，试列举主要的产品名称。

【解析】 主要的产品如表 2-2 所示，请读者参考。

【例题 2-8】 阅读以下关于三层 Client/Server 系统的系统分析方面的叙述，回答问题 1 到问题 3。

表 2-2　CORBA 中间件产品

产　　品	支 持 语 言	采用协议	提供服务类型	支 持 平 台
VisiBroker	IDL、C++、Java、COBOL	IIOP	Naming、Transactions、Event、Sccunity 等	Solaris、Win98/NT、Linux 等
IBM	IDL、C/C++、Java、COBOL	IIOP	Naming、Externalization 等	Win98/NT、OS/2 等
Sun	IDL、C/C++、Java	IIOP	Naming、LifeCycle、Event、Properity	Solaris、Win98/NT
IONA	IDL、C/C++、Java、COBOL、Ada	IIOP	Naming、Trading、Event、Sccunity 等	Soaris、Win98/NT、Linux 等
BEA	IDL、C++、Java	IIOP、DCE	LifeCycle、Sccurity、Transactons 等	Solaris、Win98/NT、Linux 等
HP	IDL、C++、Java	IIOP、DCE	Naming、Trading、Event、Secunity 等	Solaris、Win98/NT、Linux 等

某大型证券公司原来已采用两层的 Client/Server（以下简称为 C/S）方式实现了日常的证券业务交易和信息管理工作。随着业务的日益扩大、安全性要求的增高和交易处理的内容与范围的扩充，准备改造与升级其应用系统。比如：希望采用多种平台和接纳来自更多的数据源的业务处理，以更多的经营规则来实现联机事务处理（OLTP），使用具有不同数据库和操作系统的四类服务器，在系统中也将要集成更多的客户机与应用程序等。

公司信息管理部门准备采用三层 C/S 结构来升级开发该应用系统，他们把该公司 C/S 结构应用系统中的常规处理流程抽象地概括成为如图 2-3 的形式。

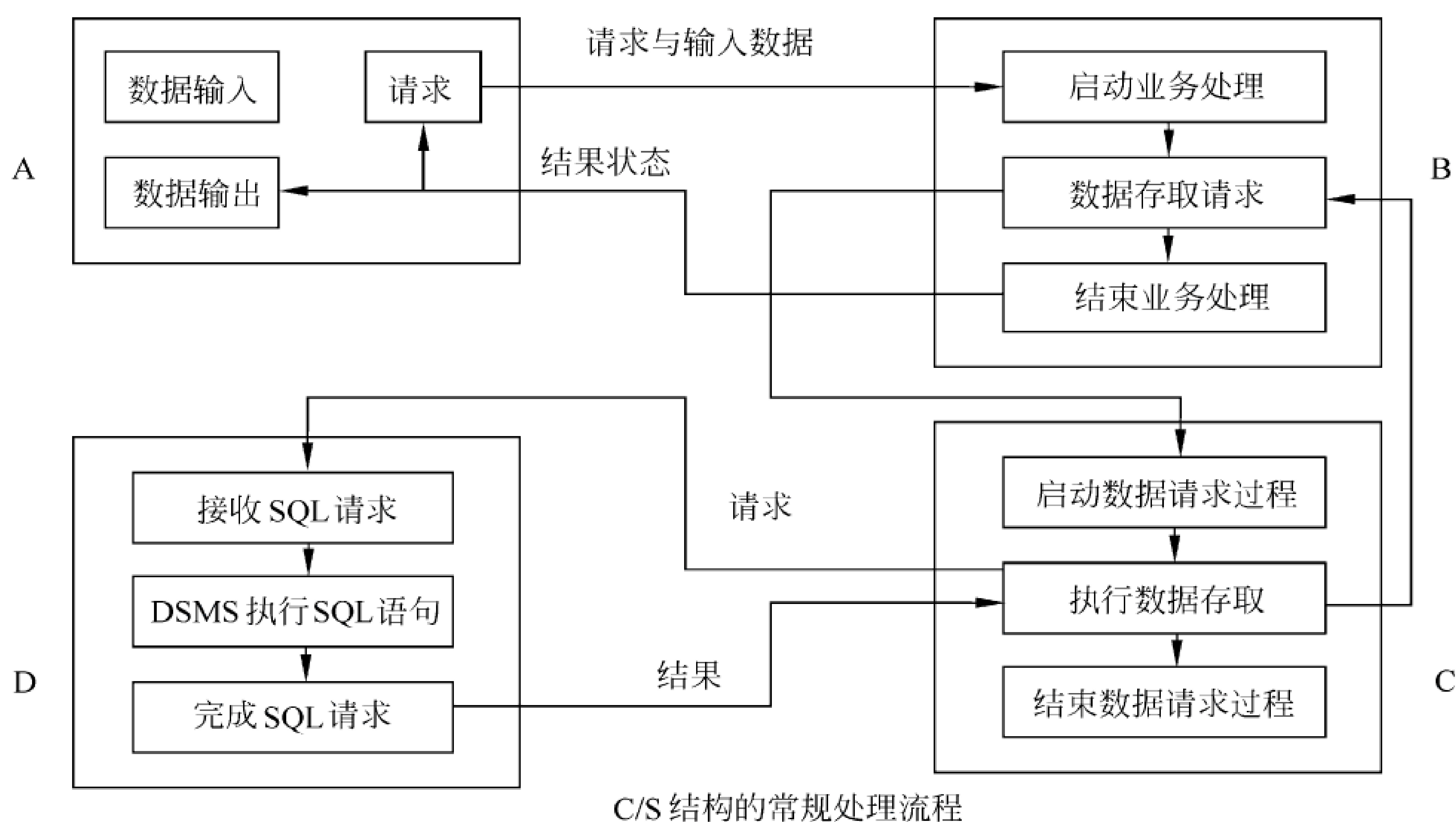


图 2-3　C/S 结构处理流程

他们打算把本公司的 C/S 网络应用系统分成 3 层，并准备采用面向对象分析与设计的方法加以实施，这 3 层大体上如下所述。

① 第一层为表示层，即该公司应用的用户接口与对话部分，比如采用 GUI 界面检查用户输入数据，显示输出的数据与信息，其中不包括公司相应业务的处理逻辑。

② 第二层是功能层，即是公司中各类业务处理具体逻辑，这是公司应用的本质性的部分。

③ 第三层是数据层，由 DBMS 承担数据库的存取与管理，比如包括公司内数据的登录、更新和检索等。

【问题 1】 信息管理部门的同事发现不论在二层 C/S 系统中，还是在三层 C/S 系统中，如图 2-1 所示的流程好像差不多。请在 100 字以内，简要地说明基于瘦客户机的三层 C/S 系统与原来的二层 C/S 系统相比，对图 2-2 中处理流程的功能划分上存在的显著差别（可用 A、B、C、D 4 块进行划分）。

【解析】 对图 2-2 中处理流程的功能划分上存在如下显著的差别。

二层 C/S：客户端 A+B，服务器 C+D（也可以是客户端 A+B+C，服务器 D）。

三层 C/S：A 客户端，B+C 在功能服务器，D 在数据库服务器。

【问题 2】 发现在三层 C/S 结构中，中间件（Middleware）有着更大的作用，他们认为中间件是一类采用应用编程接口 API 定义的软件层，提供了分布系统中通信接口，并可解决各类潜在的不兼容性方面的问题。

请在 100 字以内，简明地列出该证券公司的三层 C/S 系统中可能会用到哪几类中间件（按功能列出名称即可）。

【解析】通信中间件，数据库访问中间件，事务处理中间件，分布对象中间件（和可能有的远程过程调用中间件）。

【问题 3】 他们提出该公司新的三层 C/S 系统应当实现下列一些主要目标。

① 在公司的任意客户端访问点，允许有可能访问到公司规定的任何地方的数据库；

② 采用公司内统一的用户界面，可以访问到各类信息系统；

③ 允许实现跨平台的各类互操作性，支持异种数据库；

④ 提供高可用性、高可靠性和高安全性；

⑤ 具有良好的开放型、可扩性和可升级性；

⑥ 维护方便，成本不高，有较高的性能价格比。

请在 50 字以内指出为了实现这些目标，该类系统在开发与设计时需要特别注意的也是最关键的两类问题是什么？

【解析】 最关键的两类问题如下所述。

① 通信效率（三层通信方法、通信频率与数据量）。

② 提高各层之间的独立性（减少耦合）。

【例题 2-9】 阅读以下关于构件技术方面的叙述，回答问题 1 和问题 2。

近年来，基于面向对象的“构件”（Component，也称组件）技术得到了迅速的发展，特别是在分布式网络应用环境下企业应用系统的开发和应用程序的集成已开始广泛地运用构件技术。某企业的信息部门提出以下一些看法。

① 当前有 3 类主要的分布式对象使用与管理模型，即 CORBA（公用对象请求代理结构）、COM/DCOM/COM+（构件对象模型）和 EJB（Enterprise JavaBeans）。这几类构件模型的发展，使应用软件有可能采用装配相应“构件”的方式进行开发或者集成，这些基于对象技术的“构件”是允许可装配、可复用的，并且能通过标准接口进行通信，适合于分布式环境下的应用。

② 目前市场上提供的一些可复用的构件，可以是比传统的对象类更大的功能块，比如 Axtive X 构件，OLE 构件（OCXs）等，一般认为构件是可复用的大粒度的对象。

③ OMG（对象管理团体）所提出的 CORBA 是最早（1990 年）推出的分布式对象使用与管理模型。它的主要优点是能支持多种平台，有许多家供应厂商的支持，允许采用多种语言编写 CORBA 对象。缺点是由不同厂商所提供的产品存在互操作性问题，使用与管理过于复杂，往往要求用户对其底层有较深入的了解。

④ 微软公司推出的 COM/DCOM/COM+在当前有着数量最多的用户，这是由于其开发工具容易使用，可以用多种语言开发，并且广泛适用于多种平台。

⑤ Sun 公司推出的 EJB（Enterprise JavaBeans）基本上建立在 Java 语言基础之上，使用相对简便，能支持多种平台，并且已经有了最为丰富的第三方开发工具和产品的支持。

⑥ 微软的 NT 4.0 提供了 DCOM 有关的部件，Windows 2000 提供了 COM+有关的部件，即如同 Windows 98 捆绑式免费提供 IE 浏览器一样，DCOM/COM+部件也是免费捆绑式提供的。

⑦ 微软推出的开发工具如 Visual Studio 的最新版本等可以用来简化 COM/DCOM/COM+模型的应用，可以相对容易地建立起 COM 构件。

⑧ 采用同样的一批可复用的构件作为底层，有可能使开发出来的若干个相对独立的应用软件，组合成为一个应用软件系列（族）。正如微软 Office 套件是一个应用软件系列，其中包括了 Word、Excel、Access、Powerpoint 等若干个相对独立的应用软件。

⑨ 在开发或集成应用软件时，可以使用现成的构件库。用户在一个构件库中复用某些构件时，即是从构件库中输出被挑选出来的类型、类、关系以及附属的文档等。

⑩ 构件库或者构件系统作为产品，具有通用性和可复用性。通用性指的是其中的每一个构件都应当有广泛的适用性，即不需要任何客户化的工作就可以立即使用于各类应用软件中。

作为一个开发软件的单位，通常需要使用多个构件库或构件系统，也可以自行设计新的可复用的构件，在设计可复用的构件时，应当尽量降低可复用构件之间相互的依赖性。

【问题 1】 在上述 11 条叙述中有 3 条是不正确或不恰当的，请指出其序号，并各用 50

字简要说明理由。

【解析】 3 条不正确或不恰当的叙述如下所示。

第 4 条，只能用于 Microsoft 平台（对其他平台支持很少）。

第 5 条，EJB 的主要缺点是缺乏第三方开发工具和产品支持。

第 10 条，许多构件为了实现通用性，具有“可变性”，即针对不同应用须进行专门化和客户化。

【问题 2】 该企业的信息部门组织人力采用 COM+技术初步开发了本企业基于三层 Client/Server 模式的企业管理信息系统，在运行一段时间后，又用 COM+技术改进一个订单输入应用程序，其中有管理业务的功能（如货运费计算），又有在网络环境下导航的程序（如访问库存数据库），并且把该应用集成到原有的企业管理系统中。

请在 100 字以内简要说明采用这类构件方式进行企业应用集成的好处是什么？

【解析】 采用这类构件方式进行企业应用集成有如下好处。

① 能在构件级共享整个企业的业务逻辑和应用服务；

② 能消除重复的逻辑，优化处理过程；

③ 适应环境变化，开发集成的成本低、周期短；

④ 应用服务的可管理性好。

【例题 2-10】 阅读以下关于应用服务器分析方面的叙述，回答问题 1 到问题 3。

某软件公司已从事过不少基于 Web 的数据库应用系统的开发工作，在这些系统中都采用了三层 Client/Server 结构。在各类应用中曾选用过的 Web 服务器有：Windows NT 或 Windows 2000 相应的 IIS，Linux 系统相应的 Apache，iPlanet 的 iWS（即原 Netscape 的 Web Server）等。

公司的总工程师对几年来有关的开发项目进行了一次总结，认为在本公司开发项目中曾存在过不少问题，比如：最早是基于 Web 服务器所提供的扩展接口（如 CGI 与 API 等），使用 C 语言或 Perl 语言等编写具体应用。这类开发方式对编写应用程序的程序员要求很高，开发周期较长，调试较为困难。近两年来，公司业务很忙，大多又采用了基于服务器端的脚本语言进行开发，比如相应地使用了 ASP，PHP 或 JSP 等，虽然开发的速度显著提高，但是从严格要求来看，所开发的应用系统有如下潜在的缺陷。

① 可扩性尚不理想；

② 安全性与高可用性考虑较少；

③ 系统集成不够方便；

④ 在性能上有待提高等。

因此，总工程师最近已要求各开发组尽可能采用当前国际上主流的“应用服务器”体系结构，在条件许可的企业应用项目中选用合适的基于 Web 工作方式的应用服务器的优秀产品，可能时还应努力尝试采用规范化的 J2EE（Java 2 企业版）平台。

为了帮助各个开发组更好地理解与选用 Web 应用服务器，总工程师组织了多次讨论，

根据本公司的实践，大家普遍认为所选用的 Web 应用服务器必须强调以下 4 个方面的特征。

① 必须很好地支持对象组件（Component），提供清晰的组件工作与管理模型；

② 有良好的基于分布计算方式的管理能力，如支持多个应用服务器运行，提供负载均衡功能，实施服务器故障转移等；

③ 充分重视应用服务器部署的速度和易用程度；

④ 高效地支持对后端数据库服务器的访问，比如采用数据库连接池等。

然而，在讨论到具体技术问题时，不少软件工程师也出现了一些不同的看法。

【问题 1】 应用服务器的具体实现中，处理负载大体上可分为基于进程工作方式与基于线程工作方式两大类型。比如基于进程方式可以预先生成好所有的应用服务器进程，应用服务器在收到请求时，由对应的某个进程完成此请求的有关处理。

请在 100 字以内简要地列举出与基于线程工作方式相比，基于进程方式处理的主要弱点是什么?

【解析】 基于进程方式处理主要有如下弱点。

① 未能充分利用“线程级并行性”（或者说并发处理能力较弱）；

② 占用更多的地址空间资源。

【问题 2】 在讨论中普遍认为，各个 Web 应用服务器在嵌入对象的方式上可能会有相当大的差异，这主要反映在对象放置的位置和对象采用的接口标准上。比如对象可以放置在应用服务器上，也可以放置在应用服务器的后端。

请在 50 字以内简要列举出当前已经成为标准的对象访问接口的名称。

【解析】 CORBA、DCOM 与 EJB。

【问题 3】 对于在分布系统中会话管理的方式，也引起了热烈的讨论。通常每个 HTTP 请求需要进行一次 TCP 连接，一般可采用 Cookie、IP 地址识别技术等方式实施会话管理，从而方便系统的使用（比如允许用户登录后，系统能记住用户的基本信息等）。

但是，在多个应用服务器场合，会话信息存放的地点可以采用多种方式。比如在每一个应用服务器上保存该服务器所对应的会话信息是一种方式；设立专用的一个会话服务器来保存所有的会话信息则是另一种方式。

请在 100 字以内简要列举设立专用的会话服务器的方式有什么优点与缺点。

【解析】 设立专用会话服务器方式的优点是便于实现负载分配算法，可加强容错能力；缺点是附加一次网络通信时间，导致处理速度略为降低。

【例题 2-11】 阅读以下关于企业信息集成和应用集成方面的叙述，回答问题 1 到问题 3。

某个专门从事 IT 产品的信息报道与网上交易业务的垂直型电子商务市场，经过一年多运营已初步积累起一定的经验，在同行业中开始有了一定的知名度。

该市场的信息系统原来由“产品信息发布”、“产品动态报价”、“网上交易”和“网上财务资金结算”等子系统所组成，各个子系统的工作相对已比较稳定。但是也发现了信息

来源过于分散，数据有时不大一致，不少信息利用效率相当低等一些问题。

市场的信息技术主管在分析了该市场的运作情况后，认为应当根据国外先进的电子商务市场的经验与技术，进一步开发一个“信息集成系统”。

① 信息集成系统的主要目标

对本市场目前已涉及到的有关 IT 产品各种来源的基本数据、信息与有关资料进行集成化管理，并且能提供相应的网上服务，其目标是逐步建立起有关 IT 产品的信息交流、服务、辅助监控管理以及决策分析的一个相对完整的体系。

提高市场的服务质量和管理水平，吸引更多的供应商与采购者进入并使用本市场。

② 新信息集成系统的总体构架

充分利用该市场原有的开放式的异构平台，采用 UNIX 或 Linux 操作系统，在充分发挥其作用的基础上加以扩充与升级。

基于 Internet/Intranet 的网络体系结构，有保证系统可扩性、可靠性、安全性等方面的相应措施。

实现强有力的 OA 办公自动化系统和邮件系统，能分别规范好市场管理层、市场内部使用层和市场外部用户进行信息共享的级别与权限。

基于某个名牌的 DBMS（如 DB2，Oracle，Sybase 等）构建集成化的 MIS 系统，并且进一步采用基于 SAS 的数据统计、分析与决策系统。

根据总体构架要求，系统将由“OA 系统”、“集成化的 MIS 系统”、“数据处理与分析系统”和“网络管理系统”4 大部分所组成。

【问题 1】 技术主管指出在原来的系统中，某些 IT 产品的价格行情报表以及统计分析数据与图形的实时性太差，应当充分利用强大的数据库的能力，真正实现“动态”显示的功能。

你认为应当采用哪些主要技术？请在 50 字以内，简要列举相应技术的名称。

【解析】 Java Servelet，JSP，JDBC 和 Applet 等。

【问题 2】 通过市场内管理人员、技术人员和协作的软硬件公司的共同努力，经过 3 个月的艰苦奋斗，初步建立了一个“信息集成系统”的原型，在此原型中已采用了 SAS 软件作为统计分析数据处理的主要平台。但是，技术主管认为目前在 SAS 平台上的应用程序能力远没有得到充分的发挥。

请在 100 字以内简要指出，原型系统未能发挥 SAS 应用能力的根本原因是什么?

【解析】 SAS 必须建立在大量历史数据基础上，只有已建成的 DBMS 构架充分积累有效数据后，才能发挥更大作用。

【问题 3】 信息对于电子商务垂直市场是至关重要的，信息的集中与有效管理对于信息的再利用更是十分关键的，技术主管认为在搞好信息集成的同时应该重视应用的集成，他指出有可能采用多种方式进行本市场内各类应用的集成。比如以下所述的集成方法。

① 采用多种“数据连接器”（Connectors），在原有的常规构架基础上使用传统方式进

行数据集成，即由数据源提取出数据，经过变换与处理，然后用于更新目标数据。

② 在应用接口级上连接业务处理的过程与数据，即由开发人员通过定制或套装，经由应用程序所提供的应用接口实施应用集成。

③ 在组件（Component）级共享整个市场内的业务逻辑。事实上是使各类“应用服务”尽可能地被共享。

④ 利用“用户界面”作为集成的基础。这时，开发人员可通过对用户界面的程序化的访问，使相应的业务处理过程连接在一起，充分利用现有的处理逻辑。

如果该市场已选用了一个优秀的基于“组件”（构件）的开发环境，为了使该市场中不同的应用可以透明地进行通信和访问共享信息，请在 100 字以内，简要地以提纲方式列举出该市场实现“应用集成”的基本步骤。

【解析】 实现“应用集成”的基本步骤如下所述。

① 系统平台的集成；

② 数据的集成（如不同数据库系统之间转换数据）；

③ 应用的集成（如组件的装配，由中间件提供帮助）；

④ 业务处理流程的集成（如采用某种工作流方式）。

2.2 设计输入输出

屏幕界面设计应具有可使用性、灵活性、复杂性和可靠性等。设计输入输出检查方法和检查信息是输入、输出的关键点。与数据库交互与连接的方法有多种，不同的编程语言和开发环境（如 C 语言，以及 Java、Visual Basic、Visual C++、PowerBuilder、Delphi 等）与数据库互连的方法也不一样，常用的包括 ODBC、DAO、RDO、ADO、JDBC、JDO 等连接方法。

如图 2-4 所示是本节的知识框图。

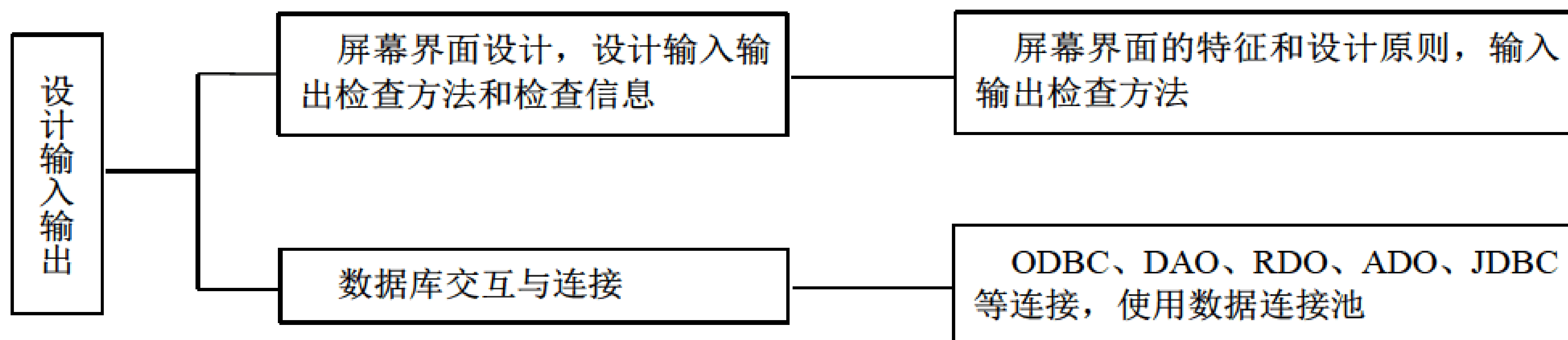


图 2-4 设计输入输出知识框图

1．知识点提炼

（1）屏幕界面设计，设计输入输出检查方法和检查信息

屏幕界面，即通常所说的用户界面，应该具备以下的特性。

- 可使用性。可使用性包括使用的简单性、用户界面中的术语标准化和一致性、拥有帮助功能、快速的系统响应和低的系统成本、一定的容错能力。
- 灵活性。灵活性包括算法的可隐可显性，允许用户根据需要制定和修改界面显示方式，能够按照用户的希望和需要提供不同详细程度的系统响应信息，与其他软件系统基本一致的标准界面。需要注意的是：为使屏幕界面具有一定的灵活性，需要付出代价，甚至有可能降低软件系统的运行效率。
- 复杂性和可靠性。屏幕界面的复杂性是指界面的规模和组织的复杂程度。在完成预定功能的前提下，屏幕界面应该越简单越好。屏幕界面的可靠性是指界面无故障使用的时间间隔，屏幕界面应能保证用户正确、可靠地使用系统，保证有关程序和数据的安全性。

在屏幕设计时，应注意以下几点。

- 通常把屏幕划分为数据输入、命令和出错处理 3 个区域，要用不同的底色来区别各个区域；
- 标题、命令、重要的提示和填充指令应是简练、准确的，应使用为用户易于理解的词汇；
- 当回答中包含的字符数已知时，数据输入区应设置有相应格式的回答区域；
- 数据输入区内各个输入项应左侧对齐。在空间允许时，最好一行仅对应一个输入；
- 如果输入中有量的单位时，单位应在输入项中的左边指定；
- 界面显示应尽量少使用代码和缩写；
- 提供简明的标题以及提示信息，便于用户浏览各种显示画面；
- 采用颜色、字符大小、下划线或不同的字体等方式来强化重要数据。
- 用户界面的设计包括输入设计、输出设计、操作设计和使用手册，如图 2-5 所示。

输入界面设计的目标是尽量简化用户的工作，尽可能地减少输入的出错率。因此，设计输入界面时要考虑尽可能减少用户的记忆负担，使界面具有预见性和一致性，防止用户输入出错，尽可能增加数据自动输入。

输入的数据内容包括数据项名称、数据内容、精度和数值范围等。输入设计的内容包括：确定输入数据内容、选择数据输入形式、设计数据的输入格式、设计输入的检查方法和检查信息。

输入设计应该遵循如下几个原则。

- 最小量原则。保证满足处理要求的前提下使输入量最小。
- 简单性原则。输入的设备、输入过程应尽量简单。
- 早检验原则。对输入数据的检验要尽量接近源数据发生点，使错误能够及时得到改正。
- 少转换原则。输入数据应该尽量使用其处理所需的形式记录，以免数据转换时发生错误。

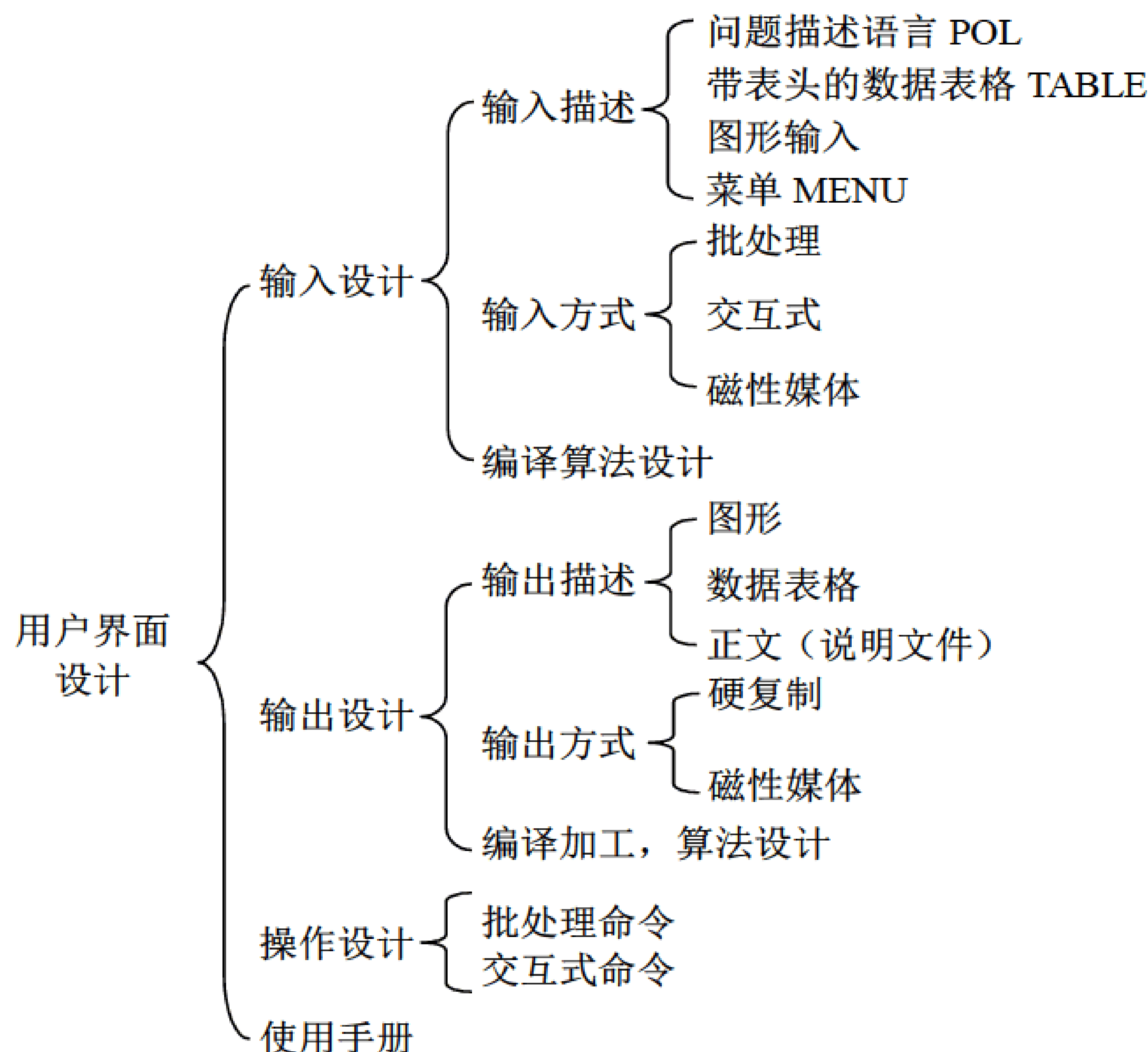


图 2-5　用户界面设计

要减少用户输入的工作量，可以采取如下的措施。

① 对于共同的输入内容设置默认值。

② 使用代码和缩写。

③ 自动填入已输入过的内容或需要重复输入的内容。

④ 如果输入内容来自一个有限的备选集，可以采用列表选择方式。备选集中的数据内容应当根据它们的使用频率，或者它们的重要性，或者它们的输入次序进行组织。

⑤ 输入屏幕应当与输入格式相匹配。实际设计数据输入（特别是大批量的数据统计报表输入）时，常常遇到统计报表（文件）结构与数据库文件结构不完全一致的情况，如有可能，应尽量改变统计报表或数据库关系表二者之一的结构，使其一致，以减少输入格式设计的难度。现在还可采用智能输入方式，由计算机自动将输入送至不同表格。

常见的数据输入方式有菜单选择输入、关键词数据输入、条形码（Bar Code）、声音数据输入、网络数据传送、光学标记 / 识别（OMR）、磁性墨水字符识别（MICR）等。数据输入的具体形式需要根据总体设计和数据库设计的要求来确定。通常在设计新系统的输入方式时，应尽量利用已有的设备和资源。

数据输入很容易出错。出错的原因可能是忽略了某一项，或在某一项的输入中键入了不正确的数据，或是数字、字符键入错误。数据验证是检查是否所有必需的项目都已填充，数据输入是否正确、是否合理。输入设计一定要考虑适当的校对措施，以减少出错的可能性。常用的校对方式有人工校对、二次键入校对（同一批数据两次键入）和数据平衡校对。

数据校对可能得到 3 种结果，分别如下所述。

- ❑ 致命错误。致命错误是指引起处理混乱的错误。此时，用户要么重新输入一个正确的数据，要么退出输入。
- ❑ 警告。警告是由很不可信的数据引起的错误，此时应停止处理并提请用户重新输入数据。
- ❑ 建议。建议是由不太可信的数据引起的错误。此时，处理不必停止，但要发出一个警告信息，使得用户或是立即停止检查，或是在处理结束时进行检查。

输出设计包括确定输出内容，选择输出设备与介质，确定输出格式，具体如下所述。

确定输出内容。根据用户需求，设计输出信息的内容。输出信息的内容包括信息形式、输出项目、数据结构、数据类型、位数及取值范围、数据的生成途径、数据的完整性和一致性的考虑等。

选择输出显示的内容，应当考虑以下几个准则。

① 只显示必需的数据，与用户需求无直接关系的数据一律不显示。

② 将要显示的数据进行分组，将每组数据按一定的结构形式来安排，相关联的数据显示在一起。

③ 输出显示的数据应与用户执行的任务有关。

选择输出设备与介质。常用的输出设备有显示终端、打印机、绘图仪和多媒体设备等，输出介质有纸张、磁带光盘和多媒体介质等。具体使用中，应该根据用户对输出信息的要求，并结合现有设备和资金条件选择输出设备和介质。

确定输出格式。输出格式应做到格式清晰、美观、易于用户阅读和理解。最终输出方式有 3 种：报表输出、图形输出和文字输出。具体采用哪种输出方式，应根据系统分析和管理业务的要求来确定。一般来说，对于基层和具体事务的管理者，应用报表方式给出详细的记录数据为宜。而对于高层领导或宏观、综合管理部门，则应该使用图形方式用以显示综合数据或发展趋势等信息。与其他两种输出方式相比较，文字输出不直观，一般较少采用。

（2）数据库交互与连接

在脚本程序中连接数据库一般都需要相应的接口来完成。连接数据库的常用方法有：ODBC、DAO、RDO、ADO、JDBC、JDO 等。

① ODBC。ODBC（Open Database Connectivity）即开发式数据库连接，是 Microsoft 公司开发的一套统一的程序接口，通过该接口可以存取不同厂商生产的数据库。ODBC 基于 SQL（Structured Query Language），并把它作为访问数据库的标准。ODBC 是一种用来在相关或不相关的数据库管理系统（DBMS）中存取数据的，用 C 语言实现的，标准应用程序数据接口。通过 ODBC API，应用程序可以存取保存在多种不同数据库管理系统（DBMS）中的数据，而不论每个 DBMS 使用了何种数据存储格式和编程接口。这个接口提供了最大限度的相互可操作性，一个应用程序可以通过一组通用的代码访问不同的数据

库管理系统。

ODBC 的结构包括 4 个主要部分：应用程序接口、驱动器管理器、数据库驱动器和数据源。

应用程序接口用来屏蔽不同的 ODBC 数据库驱动器之间函数调用的差别，为开发人员提供统一的 SQL 编程接口。

驱动器管理器为应用程序装载数据库驱动器。ODBC 管理器负责安装驱动程序，管理数据源，并帮助程序员跟踪 ODBC 的函数调用。在 ODBC 中，应用程序不能直接存取数据库，它必须通过管理器与数据库交换信息。

数据库驱动器实现 ODBC 的函数调用，提供对特定数据源的 SQL 请求。如果需要，数据库驱动器将修改应用程序的请求，使得请求符合相关的 DBMS 所支持的文法。

数据源由用户想要存取的数据以其相关的操作系统、DBMS 和用于访问 DBMS 的网络平台组成。

ODBC 是最早的整合异质数据库的数据库接口，获得极大的成功，现在已成为一种事实上的标准。访问数据库最常用的方法就是通过 ODBC。ODBC 技术是后来发展的 DAO、RDO 及 ADO 等数据库访问技术的基础。

② DAO。DAO（Data Access Objects）即服务器访问对象，是 Microsoft 公司开发的一套主要应用程序及开发工具，用它可以访问数据库的标准对象，如 Access、VB、Excel、Word 等。

③ RDO。RDO（Remote Data Objects）即远程数据库访问对象，是 Microsoft 公司为增强 DAO 的功能而推出的新产品。该产品强化了 SQL Server 的访问功能，提高了它的执行效率。

④ ADO。ADO（ActivteX Data Objects）即 ActivteX 数据对象，是 Microsoft 在 Internet 领域采取的新举措。ADO 是一组优化的访问数据库的专用对象集，它本身并不是一项新技术，而是汲取了 DAO 和 RDO 的精华，成为一个更适合于 Internet 的小而精的对象群。ADO 实际上是脚本程序连接数据库的一种选择，它为 ASP 提供了完整的站点数据库解决方案，它作用在服务器端，提供含有数据库信息的主页内容，通过执行 SQL 命令，让用户在浏览器画面中输入、更新和删除站点数据库的信息。

ADO 主要包括 Connection、Recordset 和 Command 3 个对象。Connection 对象负责打开或连接数据库文件；Recordset 对象负责存取数据库的内容；Command 对象负责对数据库下达行动查询指令，以及执行 SQL Server 的存储过程。

⑤ JDBC。JDBC（Java Database Connectivity）即 Java 数据库连接，是随着 Java 的广泛使用而发展的，能使 Java 应用与各种不同数据库对话的方式，可用于执行 SQL 语句的 Java API（Application Programming Interface）即应用程序设计接口。它由一组用 Java 语言编写的类与接口组成。JDBC 负责同一个数据库建立连接，向数据库发送 SQL 语句，并处理数据库返回的结果。JDBC 不仅支持两层模型，也支持三层模型访问数据库。JDBC 为数

据库应用开发人员、数据库前台工具开发人员提供了一种标准的应用程序设计接口，使得开发人员可以用纯 Java 语言编写完整的数据库应用程序。JDBC 2.0 API 由 JDBC 2.0 核心 API 和 JDBC 2.0 标准扩展 API 两部分组成。JDBC 2.0 核心 API 包含在 Java.sql 里面。

⑥ JDO。JDO 是 Java 对象持久化的新规范，试图提供一个对象持久化的完全版本。JDO 对象模型基本上是 Java 的对象模型，包括所有的基本类型、引用、集合和事件接口。JDBC 仅支持 RDBMS（关系型 DBMS），而 JDO 可以处理任何类型的数据源，包括 RDBMS、ODBMS、TP 监控处理、ASCII 无格式文件、xml 文件、properties 文件和大型机上的 Cobol 数据库等。

JDO 已经整合入 J2EE 体系。JDO 依靠一个新的 Connector 规范来管理一个 J2EE 应用和 JDO 容器之间的交互。JDO 容器和 J2EE 应用服务器交互，得到数据源的连接，并根据应用服务器的 JTA 兼容的事务管理器执行事务。

考生应该掌握 C 程序设计语言，以及 Java、Visual Basic、Visual C++、PowerBuilder、Delphi 中任一种开发工具与数据库互连的方法。

2．难点分析

在选用界面形式的时候，应当考虑每种类型的优点和限制。从以下几个方面来考察，进行抉择。

- ❑ 使用的难易程度：对于没有经验的用户，该界面使用的难度有多大。
- ❑ 学习的难易程度：学习该界面的命令和功能的难度有多大。
- ❑ 操作速度：在完成一个指定操作时，该界面在操作步骤、击键和反应时间等方面效率有多高。
- ❑ 复杂程度：该界面提供了什么功能、能否用新的方式组合这些功能以增强界面的功能。
- ❑ 控制：人机交互时，是由计算机还是由人发起和控制对话。
- ❑ 开发的难易程度：该界面设计是否有难度、开发工作量有多大。

在数据的连接技术方面，数据库的连接池是近年来的热门考点，下面着重讲解一下数据库连接池。一般情况下，在使用开发基于数据库的 B/S 程序时，传统的模式基本是按以下步骤进行的。

① 在主程序（如 Servlet、Beans）中建立数据库连接。

② 进行 SQL 操作，取出数据。

③ 断开数据库连接。

使用这种模式开发，存在很多问题。首先，要为每一次 Web 请求建立一次数据库连接，对于一次或几次操作来讲，或许觉察不到系统的开销。但是，对于 Web 程序来讲，即使在某一较短的时间段内，其操作并发请求数也远远不是一两次，而是数十上百次，在这种情况下，系统开销是相当大的。事实上，在一个基于数据库的 Web 系统中，建立数据库连接的操作将是系统中代价最大的操作之一。很多时候，B/S 应用的速度瓶颈就在于此。

其次，使用传统的模式，必须去管理每一个连接，确保它们能被正确关闭，如果出现程序异常而导致某些连接未能关闭，将导致数据库系统中的内存泄露，最终导致重启数据库。

针对以上问题，可以使用连接池技术来解决上述问题。首先，介绍一下连接池技术的基本原理。顾名思义，连接池最基本的思想就是预先建立一些连接放置于内存对象中以备使用，当程序中需要建立数据库连接时，只须从内存中取一个来用而不用新建。同样，使用完毕后，只须放回内存即可。而连接的建立、断开都有连接池自身来管理。同时，还可以通过设置连接池的参数来控制连接池中的连接数、每个连接的最大使用次数等等。通过使用连接池，将大大提高程序效率，同时，还可以通过其自身的管理机制来监视数据库连接的数量、使用情况等。

3．典型例题

【例题 2-12】 一般来说，相同的功能在用户界面上可以有不同的表现形式，在选用界面形式的时侯，应当考虑不同表现形式的优点和限制。根据自己的项目经验，谈谈设计用户界面时应从哪些方面来考虑？

【解析】 应该从如下的几个方面来考虑。

- ❑ 使用的难易程度：对于没有经验的用户，该界面使用的难度有多大。
- ❑ 学习的难易程度：学习该界面的命令和功能的难度有多大。
- ❑ 操作速度：在完成一个指定操作时，该界面在操作步骤、击键和反应时间等方面效率有多高。
- ❑ 复杂程度：该界面提供了什么功能、能否用新的方式组合这些功能以增强界面的功能。
- ❑ 控制：人机交互时，是由计算机还是由人发起和控制对话。

开发的难易程度：该界面设计是否有难度、开发工作量有多大。

【例题 2-13】 数据输入是指所有供计算机处理的数据的输入。数据输入界面是系统的一个重要组成部分，它常占用用户的大部分使用时间。根据自己的项目经验，谈谈数据输入界面的目标是什么？

【解析】 数据输入界面的目标是尽量简化用户的工作，并尽可能地减少输入的出错率。为此，在设计时要考虑尽可能减少用户的记忆负担，使界面具有预见性和一致性，防止用户输入出错，以及尽可能增加数据自动输入。

【例题 2-14】 根据自己的项目经验，说明在软件输入输出设计的范围，可以通过哪些方法来减少用户输入的工作量？

【解析】在软件设计的范围，可以通过以下方法来减少用户输入的工作量。

① 对共同的输入内容设置默认值。

② 使用代码和缩写。

③ 主动填入已输入过的内容或需要重复输入的内容。

④ 如果输入内容是来自一个有限的备选集，可以采用列表选择或指点方式。

【例题 2-15】 数据输入很容易出错。出错的原因可能是忽略了某一项，或在某一项的输入中键入了不正确的数据，或是数字、字符敲错。数据验证要检查是否所有必需的项目都已填充，数据输入是否正确、是否合理。出错验证可能得到以下 3 种结果：致命错误、警告和建议，根据自己的项目经验，谈谈如何解决？

【解析】 致命错误：引起处理混乱的错误。此时，用户要么重新输入一个正确的数据，要么退出输入，不允许其他做法。

警告：由很不可信的数据引起的错误。此时应停止处理并提请用户重新输入数据。

建议：由不大可信的数据引起的错误。此时，处理不必停止，但要发出一个警告信息，使得用户或是立即进行检查，或是在处理结束时进行检查。

【例题 2-16】 根据自己的项目经验，除了键盘和鼠标输入方式外，一般的信息系统中还有哪些输入方式？

【解析】 一般的信息系统中，还有如下的输入方式。

光学标记/识别（OMR）在表格中使用。用户在表格的一个区域中打标记□或■，然后让表格通过一个光敏读入设备，其中用暗标记■表示“是”，用亮标记□（即未标记过）表示“否”。例如一般考试使用的“机读卡”。

光学字符识别（OCR）系统可让计算机通过模式比较来识别一些具有不同字体和大小的印刷体。首先它让字符识别系统熟悉铅字字体的特征。经过若干次尝试，使计算机系统了解这种字体的规则，并将这些规则记忆到模式匹配算法中。

磁性墨水字符识别（MICR），MICR 字体就是在银行支票上的账号和分类号所使用的字符。

条形码（Bar Code）条形码由许多粗细不等的竖线组成的标签，这些竖线条在特定位置上出现或不出现就表示某个特定的数据。条形码的代码由一个特殊的光敏装置或条形码读入器读入，读入器在横穿过条形码时挑选出暗带，并根据暗带在位置 x，$x+1$ 等处是否出现而将条形码序列翻译成数据，计算机将条形码与记录相比较以计算出商品的号码或数值。

声音数据输入有许多很明显的优点。它输入速度很快，可用于不宜使用纸张及不能使用键盘的场合。

【例题 2-17】 阅读以下有关 ADO 的叙述，回答问题 1 和问题 2。

ADO 的全名是 ActiveX Data Object（ActiveX 数据对象），是一组优化的访问数据库的专用对象集，它为数据库连接提供了完全的解决方案。

ADO 的底层是 OLE DB，不仅能访问关系型数据库，也可以访问非关系型数据库。某单位一信息项目的数据库平台采用了 SQL Server 2000 作为后端平台，前端选择了 Microsoft 公司的 Visual C++作为开发工具，采用的 Client/Server 模式。

【问题 1】 ADO 主要包括 3 个对象，它们的名称和功能主要是什么？

【解析】 ADO 主要包括 Connection、Recordset 和 Command 3 个对象，它们的主要功能如下。

① Connection 对象：负责打开或连接数据库文件；

② Recordset 对象：存取数据库的内容；

③ Command 对象：对数据库下达行动查询指令，以及执行 SQL Server 的存储过程。

【问题 2】 在项目开始的初期，项目经理要求程序员在 stdafx.h 文件里加上下面的代码：

#IMPORT"c:\program files\common files\system\ado\msado15.dll" no_namespaces rename（"EOF"，"adoEOF"）

这行代码的作用是什么？

【解析】 这行代码的作用是告诉编译器去哪里找 ADO 的库文件（本例题中为 c:\program files\common files\system\ado\msado15.dll，在不同的计算机上路径有所不同），然后说明不用 namespace。最后，为了防止常量冲突，将 EOF 更名为 adoEOF。

【例题 2-18】 阅读以下有关 ODBC 的叙述，回答问题 1 和问题 2。

ODBC 是 Open Database Connectivity 的英文简写。它是一种用来在相关或不相关的数据库管理系统（DBMS）中存取数据的，用 C 语言实现的，标准应用程序数据接口。通过 ODBC API，应用程序可以存取保存在多种不同数据库管理系统（DBMS）中的数据，而不论每个 DBMS 使用了何种数据存储格式和编程接口。

【问题 1】 数据库 ODBC 错误是数据库应用开发过程中最常遇到的，ODBC 的日志功能为开发者提供了很好的定位问题的信息。根据自己的项目经验，列举使用 ODBC 日志功能和 ODBC SDK 的帮助文件定位错误的一般步骤。

【解析】 使 ODBC 日志功能和 ODBC SDK 的帮助文件定位错误的一般步骤如下所述。

① 打开 ODBC 的日志功能；

② 运行有问题的 ODBC 程序；

③ 停止程序，关闭日志，在日志文件中查找 SQL_ERROR 字样；有时可能用到 SQL_SUCCESS_WITH_INFO；

④ 对照 ODBC SDK 的帮助文件，找到对应出错信息的说明；

⑤ 解决错误。

【问题 2】 在什么情况下使用 ODBC 日志功能和 ODBC SDK 的帮助文件很难定位错误的具体位置，请举例说明。

【解析】 例如，在 RDO 等对象中，它们的下层还是 ODBC API，但错误信息进行了封装，所以有时候很难定位错误的具体位置。

【例题 2-19】 阅读以下有关的叙述，回答问题 1 到问题 3。

Microsoft 推出的 ODBC（Open Database Connectivity）技术为异质数据库的访问提供了统一的接口。ODBC 基于 SQL（Structured Query Language），并把它作为访问数据库的标准。这个接口提供了最大限度的相互可操作性，一个应用程序可以通过一组通用的代码

访问不同的数据库管理系统。

一个软件开发者开发的客户/服务器应用程序不会被约束在某个特定的数据库之上。ODBC 可以为不同的数据库提供相应的驱动程序，它的灵活性表现在以下几个方面。

① 应用程序不会受制于某种专用的 API。

② SQL 语句以源代码的方式直接嵌入在应用程序中。

③ 应用程序可以以自己的格式接收和发送数据。

④ ODBC 的设计完全和 ISO Call-Level Interface 兼容。

⑤ 现在的 ODBC 数据库驱动程序支持 55 家公司的数据产品。

【问题 1】 在一般的数据库应用系统中，ODBC 有 4 个组成部分：ODBC 管理器（ODBC manager）、ODBC 驱动程序（ODBC Drivers）、应用程序（Application，程序）、数据源（Data Sources，数据库）。试画出它们的逻辑关系图（主要反映调用关系）。

【解析】 逻辑关系图如图 2-6 所示。

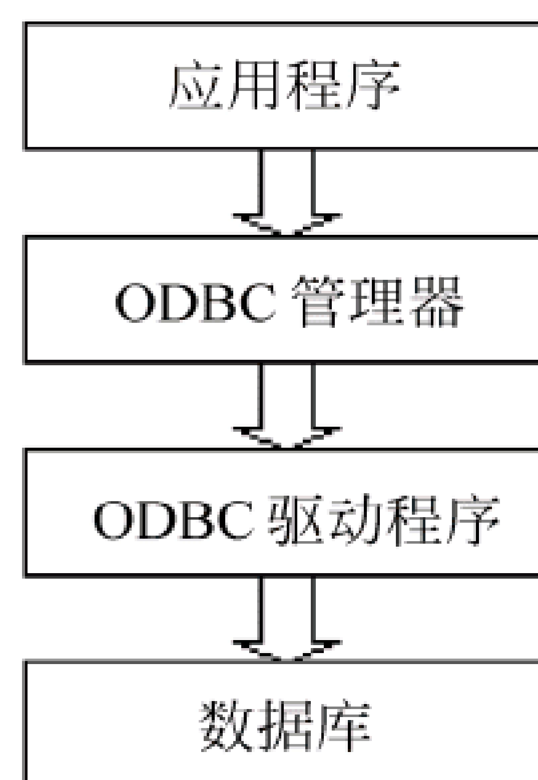


图 2-6　逻辑关系图

【问题 2】 根据自己的项目经验，说明使用 ODBC 编程的一般步骤是什么？

【解析】 按照以下几个步骤进行。

① 连接数据源。

② 创建并执行一条或多条 SQL 语句。

③ 处理结果记录（如果有的话）。

④ 断开数据源。

【问题 3】 ODBC 的 API 编程中，连接数据源时需要了解环境（Environment）和连接（Connection）的概念和用途，请说明。

【解析】 ODBC 的 API 编程中，连接数据源时需要了解环境（Environment）和连接（Connection）的概念和用途的说明如下所述。

环境（Environment）是一个全局文本用来存取数据。它包含应用于所有 ODBC 会话的信息，例如一个会话的 Connections 句柄。在用 ODBC 之前必须从环境中获得这个句柄。

连接（Connection）用于指定 ODBC 驱动程序和数据源（数据库）。用户可以在同一个环境中同时连接不同的数据库。

【例题 2-20】 阅读以下有关数据库连接的叙述，回答问题。

某单位原网络系统由 3 个 100Mbps 以太网构成，以光纤互联，服务器操作系统为 Windows NT4+SP6，数据库为 Sybase ASE12，客户端为 Windows 98，应用程序为 PowerBuilder 6.5 开发，只安装 TCP/IP 协议，使用内部固定 IP 分配。

在过去的使用过程中，一直很好。因业务发展需要，新并入一子网络，因距离过远，租用电信 128Kbps DDN 线路，出现如下故障：可以 Ping 通数据服务器，不能通过应用程序连接到数据服务器。

技术员首先用替换法将各个连接上的硬件测试了一遍，包括两端的路由器、DTU、电

信的 DDN 卡，结果一样可 Ping 通，应用程序不能连接服务器。由此可以肯定，问题出在软件上。将客户端的 Windows 98 重装，Sybase ASE 客户端重装，应用程序重装，问题依然存在。

根据提示，如果不是 Sybase 软件本身的问题，那就可能是 Windows 98 网络这一部分的问题。为什么能 Ping 通，而不能连接？难道系统只能通过光纤直接连接吗？将原来通过光纤可以连接的子网中计算机的 Windows 98 网络属性与现在通过 DDN 连接的子网的计算机的 Windows 98 网络属性对比，一模一样，没有不同。但在查看中，突然看到 Windows 98 的 WINS 配置项。选择启用 WINS，并将 WINS 服务器 IP 填入数据库服务器 IP，重启后，应用程序可以连接上数据库服务器，正常工作。

【问题】 试分析该故障的产生原因。

【解析】 过去的 3 个子网通过光纤互联，相当于在一个大的局域网中。Windows 98 虽然只安装了 TCP/IP 协议，但仍然是通过计算机名访问的方式，互相通信。当加入一个通过路由的 DDN 连接时，Windows 98 通过计算机名访问的方式不支持路由，因此只可 Ping 通，而不能启用应用程序。加入 WINS 服务后，让 IP 与机器名对应，Windows 98 才能真正启动应用程序访问数据库服务器。

2.3　设计物理数据

数据库应用的成功实施通常需要在工程的前期阶段精心设计。数据库查询效率是大多数关系数据库的设计时需要考虑的问题，在物理设计时需要确定的有：使用哪种类型的磁盘硬件或存储介质，如 RAID（独立磁盘冗余阵列）设备等；如何将数据放置在磁盘上；在访问数据时使用哪种索引设计提高查询性能等。

如图 2-7 所示是本节的知识框图。

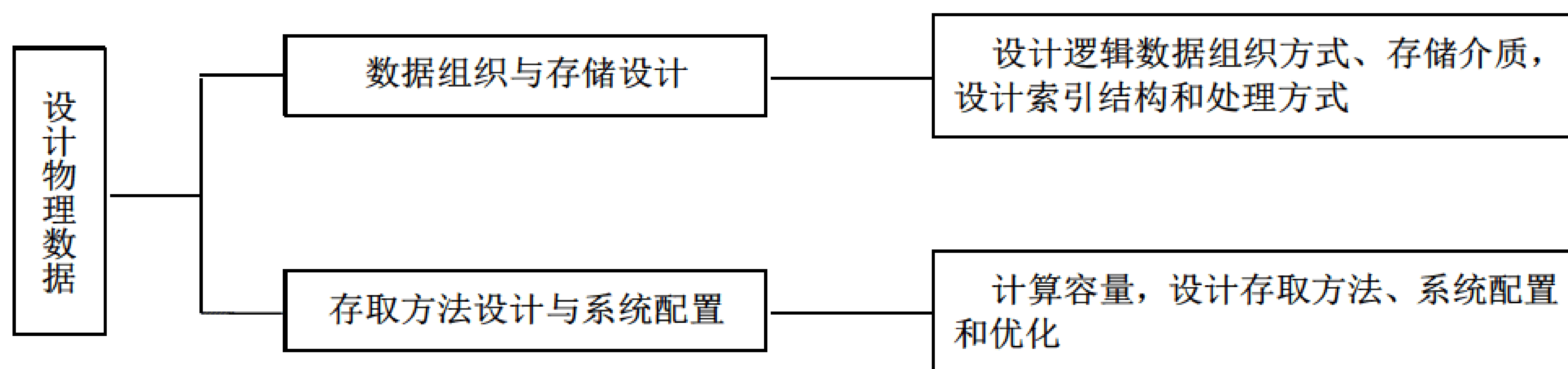


图 2-7　设计物理数据知识框图

1．知识点提炼

（1）数据组织与存储设计

为了分析事务在数据库上运行的频率和性能要求，以确定存储设备是否满足要求，系统设计人员不仅需要深入了解给定的 DBMS 功能，DBMS 提供的环境和工具、硬件环境，

特别是存储设备的特征。还应该了解应用环境的具体要求，如各种应用的数据量、处理频率和响应时间等。

在需求分析的基础上，按照需求分析中的信息要求，对用户信息加以分类、聚集和概括，建立信息模型，并依照选定的数据库管理系统软件，转换成数据的逻辑结构，再依照软硬件环境，最终实现数据的合理存储。这一过程可分解为 3 个阶段：概念结构设计、逻辑结构设计和物理结构设计。在数据库系统的概念结构设计阶段，设计人员以用户的观点，对用户信息的抽象和描述，把用户的信息要求统一到一个整体逻辑结构中，形成一个独立于任何 DBMS 软件和硬件的概念模型。逻辑结构设计是将概念设计所得到的概念模型转换为某个 DBMS 所支持的数据模型，并对其进行优化。数据库最终要存储在物理设备上。对于给定的逻辑数据模型，选取一个最适合应用环境的物理结构的过程，包括存储结构和存取方法，这是数据库物理结构设计的任务。

设计物理数据的目标是有效地实现逻辑模式，确定所采取的存储策略。应该结合具体 DBMS 的特点与存储设备特性进行设计，选择数据库在物理设备上的存储结构和存取方法。

数据库物理设计阶段的任务包括两部分：一是确定物理结构，在关系数据库中主要指存取方法和存储结构；二是评价物理结构，评价的重点是时间和空间效率。

逻辑数据组织方式确定后，设计存储记录的物理结构。在物理结构中，数据的基本存取单位是存储记录。存储记录结构包括记录的组成、数据项的类型和长度、逻辑记录到存储记录的映射。一个存储记录可以和一个或多个逻辑记录相对应。文件是某一类型的所有存储记录的集合。文件组织或文件结构是组成文件的存储记录的表示法。文件结构应该表示文件格式、逻辑次序、物理次序、访问路径、物理设备的分配。

决定存储结构的主要因素包括存取时间、存储空间和维护代价 3 个方面。DBMS 提供了聚簇和索引，可以提高数据库的存储性能。

聚簇（Cluster）就是把在一个或一组属性上具有相同值的元组集中地存放在一个物理块中，以提高查询速度。其中，这个共同的属性称为聚簇码。如果元组在一个物理块中存放不下，可以存放在相邻的物理块中。

使用聚簇有以下两大优点。

一是使用聚簇以后，聚簇码相同的元组集中存储在一个或者几个相邻的物理块，聚簇码即元组共同的属性值，不必在每个元组中重复存储，只要在一组中存储一次即可，这样可以节省存储空间，尤其是集合中元组较多的时候。

二是使用聚簇以后，可以大大提高按聚簇码进行查询的效率。只要做一次 I/O 操作，就可以获得多个满足查询条件的记录，减少了访问磁盘的次数，提高了查询效率。

存储记录是属性值的集合，主关系键可以唯一确定一个记录，而其他属性的一个具体值不能唯一确定是哪个记录。在主关系键上应该建立唯一索引，这样不但可以提高查询速度，还能避免关系键重复值的录入，确保了数据的完整性。

为了提高数据的存取速度，常采用索引技术。在数据库中，用户访问的最小单位是属

性。如果对某些非主键属性的检索很频繁，应该建立这些属性的索引文件，索引文件可以对存储记录重新进行内部链接，从逻辑上改变记录的存储位置，从而改变访问数据的入口点。建立索引文件的优点是可以缩短存取时间、提高存取效率，缺点是增加了索引文件所占用的存储空间、并且增加了维护的开销。在物理设计阶段，要从实际出发，根据数据处理和修改要求，确定数据库文件的索引字段和索引类型。

（2）存取方法设计与系统配置

将逻辑数据结构变换成物理数据结构，首先要确定数据的存储结构，其次要设计数据的存放位置、数据的存取方法和访问路径，最后还要对数据模型进行优化。

存储结构是指数据文件中记录之间的物理结构。在文件中，数据是以记录为单位存储的，存储结构可以是顺序存储、哈希存储、堆存储和 B+树存储等。在实际应用中，要根据数据的处理要求和变更频度，选定合理的物理结构。

为了提高数据的访问效率，要根据对数据的不同处理合理选择数据分布。对于使用频率高、响应时间短的数据，应存储在高速设备上。为了提高系统性能，应该根据应用情况，将数据的易变部分、稳定部分、经常存取部分和存取频率较低部分等分开存放。

存取方法是为存储在物理设备上的数据提供存储和检索能力的方法。存取方法包括存储结构和检索机构两个部分，其中存储结构限定了可能访问的路径和存储记录，检索机构则定义了每个应用的访问路径。

访问路径包括主访问路径和辅访问路径。主访问路径与初始记录的装入有关，通常是用主键来检索的。辅访问路径是通过辅助键的索引对存储记录重新进行内部链接，从而改变访问数据的入口点。用辅助索引可以缩短访问时间，但增加了辅存空间和索引维护的开销。

为了进一步提高数据库应用系统的性能，通常以规范化理论为指导，适当地修改、调整数据模型的结构，即对数据模型进行优化。

规范化理论为数据库设计人员判断关系模式的优劣提供了理论标准，可用来预测模式可能出现的问题，使数据库设计工作有了严格的理论基础，是数据模型优化的重要依据。数据模型的优化步骤如下所述。

① 确定数据依赖。

② 对于各个关系模式之间的数据依赖进行极小化处理，消除冗余的联系。

③ 按照数据依赖的理论对关系模式逐一进行分析，看是否存在部分函数依赖、传递函数依赖、多值依赖等，确定各关系模式分别属于第几范式。

④ 按照需求分析阶段得到的各种应用对数据处理的要求，分析对于这样的应用环境这些模式是否合适，确定是否要对它们进行合并或分解。

⑤ 对关系模式进行必要的分解。

2．难点分析

许多因素会影响数据库最终的大小，在估算数据库容量时必须考虑所有这些因素。以

下是需要考虑的一些主要因素。

① 每行记录的大小。每行记录都由一列或者多列数据组成。这些列决定了每一行的大小。数据库管理员需要从数据库开发人员那里或者通过自行检查数据库中的每个表得到每一行的大小。

② 记录的数量。表中的记录数有可能基本不变，也有可能变化相当大。数据库中的每个表存储多少条数据是数据库开发人员来预计的。

③ 表的数量。一些数据库只有少量的表，另一些却有可能有成百上千的表。

④ 索引的数量。每个表都有一个或者多个索引。索引分为聚簇式索引或者是非聚簇式索引两种，非聚簇式索引将在数据库中占用额外的空间。

⑤ 每个索引的大小。索引的大小取决于使用该索引的列的大小、索引中包含的记录数量和索引的填充因子。填充因子越大，索引对象占用的空间就越多。

⑥ 数据库对象的数量和大小。数据库包含很多对象，例如触发器、视图、存储过程等等。一些对象，例如存储过程，可能占用很多空间。数据库对象的确切大小由数据库开发人员确定。

⑦ 事务日志的大小。事务日志大小的差别很大，这取决于很多因素，其中包括数据库中数据的修改频率。一般来说，事务日志以数据库容量的 15%到 30%为起点，在数据库被用于实际工作环境后再通过监控来调整。因为修改比较多就意味着事务比较多，所以需要一个大一些的事务日志来存放所有这些事务，所以经常被修改的数据库比很少被修改的数据库需要更大的事务日志。事务日志的大小还受备份事务日志的频率影响。越是经常备份，事务日志就越是可以小一些，这是因为在每次备份过程中都会截断事务日志。

⑧ 数据库的计划增长量。有一些数据库的容量从不增长，也有一些每周都大幅度地增长。为了确定总体的计划增长量，管理员必须估算数据库中每个表的增长量。

3．典型例题

【例题 2-21】 阅读以下有关 Oracle 的叙述，回答问题。

某信息系统软件环境为 Windows NT 4.0 + Oracle 8.0.4；数据库管理系统 Oracle 安装路径为：C:\ORANT。

【问题】 数据库管理员使用以下 SQL 语句的作用是什么？SELECT 子句查询了数据库的哪些信息？

```
COL Tablespace_Name FORMAT a20;
SELECT
b.File_Id  File_ID,
b.Tablespace_Name  Tablespace_Name,
b.Bytes  Bytes,
(b.Bytes-Sum(NVL(a.Bytes,0))) USED,
Sum(NVL(a.Bytes,0)) Free,
```

```
        Sum(NVL(a.Bytes,0))/(b.Bytes)*100        Percent
        From Dba_Free_Space a,Dba_Data_Files b
        WHERE a.File_Id=b.File_Id
        GROUP BY b.Tablespace_Name,b.File_id,b.Bytes
    ORDER BY b.File_Id;
```

【解析】 该 SQL 语言的作用是查看各个表空间占用磁盘情况，分别查询了文件 ID 号、表空间名、字节数、已使用的字节数、剩余空间、剩余百分比。

【例题 2-22】 某单位一信息项目的数据库平台采用了 SQL Server 2000 作为后端平台，采用的 Client/Server 模式。数据库管理员要创建一个新的用于存储人员信息的数据库。从数据库开发人员那里得知每个记录的长度为 1024 字节。假设数据库中除了数据表之外没有其他对象（索引、存储过程、触发器等），那么应该创建一个多大的数据库才能最终容纳 1 000 000 条记录？（以兆为单位）

【解析】 数据库中的每兆有 128 个页面，每个页面共有 8 192 个字节，每个页面有 8 060 个字节可以被使用（由于存在页面的管理开销）。因为每个记录需要 1 024 字节，所以每个页面只能包含 7 条记录。要容纳 1 000 000 条记录，需要创建 1 000 000÷(128×7)= 1 116MB 的数据库。

【例题 2-23】 阅读以下有关系统事务和并行性的叙述，回答问题 1 到问题 4。

通常在进行数据库的新增、修改、删除、查询的时候，如果面对的不是多个用户及时单机处理的时候，一般不需要考虑数据库的表锁定以及死锁之类情况。但是如果面对的是多用户并行处理的网络环境的时候，对表锁定的问题就需要较为仔细的分析和考虑。

某单位一信息项目的数据库平台采用了 SQL Server 2000 作为后端平台，前端选择了 Microsoft 公司的 Visual Basic 作为开发工具，采用的 Client/Server 模式。在程序中，两个用户同时保存新增的数据，程序开始是这样处理的：

```
cn.BeginTrans
cn.EXECUTE "INSERT INTO TABLE A …"
SET RS = cn.EXECUTE("SELECT COUNT(*)FROM TABLE A WHERE …")
IF RS.RecordCount > 0 THEN
    '表 A 的字段 A 不能重复
    cn.RollbackTrans
ELSE
    cn.CommitTrans
END IF
```

【问题1】 程序运行后，发现系统的并行性很低，试分析该程序段并行度低的原因（SQL Server 2000 中的锁粒度采用的是表级锁）。

【解析】 当 SQL Server 在执行 INSERT 命令时，如果不添加任何参数，数据库默认申

请一个 IX 锁给表 A，当第一个用户执行 cn.EXECUTE "INSERT INTO TABLE A …" Connection 时，向数据库申请了一个 X 锁给表 A；与此同时当第二个用户执行 cn.EXECUTE "INSERT INTO TABLE A …" Connection 时，也向数据库成功地申请了一个 X 锁给表 A；但是当执行 SET RS = cn.EXECUTE（"SELECT COUNT（*）FROM TABLE A WHERE …"）这一句的时候就会有问题产生。假设第一个用户先一步执行，由于 SELECT 命令需要向数据库申请一个 S 锁给表 A，但是由于这时候表 A 已经存在一个 X 锁并且属于另外一个连接，因此只好在此等候。紧接着第二个用户也执行 SET RS = cn.EXECUTE（"SELECT COUNT（*）FROM TABLE A WHERE …"），也会向数据库申请一个 S 锁给表 A，这时候数据就会自动结束较晚申请 X 锁的连接，同时回滚这个事务。

【问题 2】 现给出解决办法：增加一张辅助的表来提高系统并行性，试写出相应的程序段。

【解析】 相应的程序段如下所示。

```
cn.BeginTrans
cn.EXECUTE "UPDATE TMPLockTable SET FieldLock=1"
cn.EXECUTE "INSERT INTO TABLE A WITH(Tablock) …"
SET RS = cn.EXECUTE("SELECT COUNT(*)FROM TABLE A WHERE …")
IF RS.RecordCount > 0 THEN
    '表 A 的字段 A 不能重复
    cn.RollbackTrans
ELSE
    cn.CommitTrans
END IF
```

【问题 3】 现给出另一种解决办法：设置数据参数让程序可以读取没有提交的数据来提高系统并行性，试写出相应的程序段。

【解析】 相应的程序段如下所示。

```
cn.BeginTrans
cn.EXECUTE "SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED "
cn.EXECUTE "INSERT INTO TABLE A …"
SET RS = cn.EXECUTE("SELECT COUNT(*)FROM TABLE A WHERE …")
IF RS.RecordCount > 0 THEN
    '表 A 的字段 A 不能重复
    cn.RollbackTrans
ELSE
    cn.CommitTrans
END IF
cn.EXECUTE "SET TRANSACTION ISOLATION LEVEL READ COMMITTED"
```

【问题 4】 现给出第三种解决办法：设置 INSERT 命令参数 WITH（Tablock）来提高系统并行性，试写出相应的程序段。

【解析】 相应的程序段如下所示。

```
cn.BeginTrans
cn.EXECUTE "INSERT INTO TABLE A WITH(Tablock) …"
SET RS = cn.EXECUTE("SELECT COUNT(*)FROM TABLE A WHERE …")
IF RS.RecordCount > 0 THEN
    '表 A 的字段 A 不能重复
    cn.RollbackTrans
ELSE
    cn.CommitTrans
END IF
```

2.4 设计安全体系

在数据库应用系统设计中，安全体系的设计是一项重要的内容，尤其是那些包含有重要商业信息的数据库项目。在设计数据库应用的安全体系中，首先需要明确安全等级的概念，以及如何划分和管理。在大型数据库访问过程中，用户要经过多个安全性阶段，如身份验证和权限验证等。身份验证主要表明用户连接数据库服务器的能力；然后用户需要访问服务器上数据库的权限，为此须授予每个数据库中映射到用户登录的账户访问权限。

如图 2-8 所示是本节的知识框图。

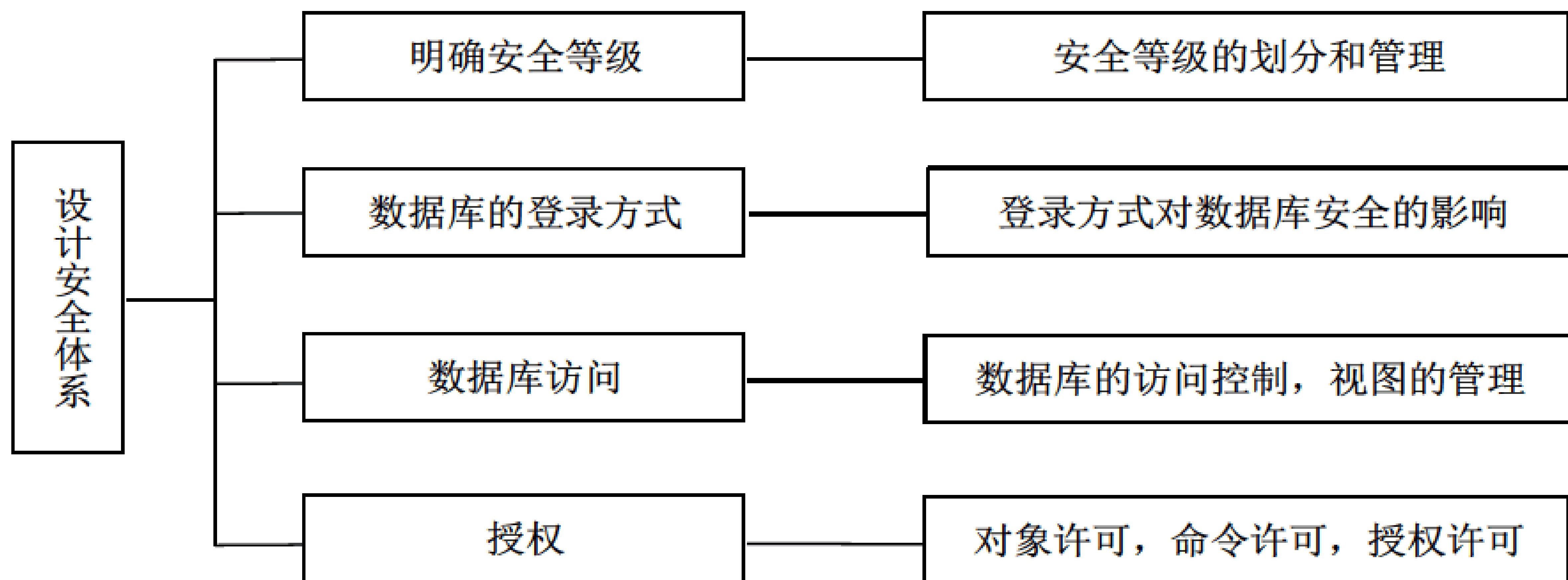


图 2-8　设计安全体系知识框图

1. 知识点提炼

（1）明确安全等级

数据库的安全性是指保护数据库，以防止非法使用所造成数据的泄露、更改或破坏。安全性控制是指要尽可能地杜绝所有可能的数据库非法访问。用户非法使用数据库可以有

很多种情况，例如，编写合法的程序绕过 DBMS 授权机制，通过操作系统直接存取、修改或备份有关数据。

实际上，安全性问题并不是数据库系统所独有的，所有计算机系统都存在这个问题。

在数据库系统中，安全措施是一级一级层层设置的。当用户进入数据库系统时，系统首先根据输入的用户标识进行身份鉴定，只有合法的用户才准许进入系统。对已进入系统的用户，DBMS 还要进行存取控制，只允许用户进行合法的操作。DBMS 是建立在操作系统之上的，安全的操作系统是数据库安全的前提。操作系统应能保证数据库中的数据必须由 DBMS 访问，而不允许用户越过 DBMS，直接通过操作系统访问。数据最后通过加密的形式存储到数据库中。

（2）数据库的登录方式

数据库系统是不允许未经授权的用户对数据库进行操作的。当用户进入数据库系统时，系统首先要根据输入的用户标识对其进行身份鉴定，只有合法的用户才准许进入系统。

用户标识和鉴定（Identification and Authentication）是数据库系统提供的最外层的安全保护措施，其方法是由系统提供一定的方式让用户标识自己的名字或身份，系统内部记录着所有合法用户的标识，每次用户要求进入系统时，由系统进行核实，通过鉴定后才提供计算机的使用权。

用户标识和鉴定的方法有多种，为了获得更强的安全性，往往是多种方法并举，常用的方法有以下几种。

① 用一个用户名或用户标识符来标明用户的身份，系统以此来鉴别用户的合法性。如果正确，则可进入下一步的核实，否则，不能使用计算机。

② 用户标识符是用户公开的标识，它不足以成为鉴别用户身份的凭证。为了进一步核实用户身份，常采用用户名与口令（PassWord）相结合的方法，系统通过核对口令判别用户身份的真伪。系统有一张用户口令表，为每个用户保持一个记录，包括用户名和口令两部分数据。用户先输入用户名，然后系统要求用户输入口令。为了保密，用户在终端上输入的口令不显示在屏幕上。系统核对口令以鉴别用户身份。通过用户名和口令来鉴定用户的方法简单易行，但该方法在使用时，由于用户名和口令的产生和使用比较简单，也容易被窃取，因此还可采用更复杂的方法。

（3）数据库访问

对于已经进入系统的用户，DBMS 还要进行存取控制，只允许用户进行合法的操作。

用户存取权限指的是不同的用户对于不同的数据对象允许执行的操作权限。存取权限由两个要素组成，即数据对象和操作类型。定义一个用户的存取权限就是要定义这个用户可以在哪些数据对象上进行哪些类型的操作。在数据库系统中，每个用户只能访问他有权存取的数据并执行有权使用的操作。因此，必须预先定义用户的存取权限。对于合法的用户，系统根据其存取权限的定义对其各种操作请求进行控制，确保合法操作。

常采用视图机制，通过为不同的用户定义不同的视图，从而限制各个用户的访问范围。

视图机制可以把要保密的数据对无权存取这些数据的用户隐藏起来，从而对数据提供一定程度的安全保护。

需要注意视图机制的主要功能在于它提供了数据库的逻辑独立性，但是它的安全保护功能往往不能满足应用系统对安全性的要求。在实际应用中，通常将视图机制与授权机制结合起来使用，首先用视图机制屏蔽一部分保密数据，然后在视图上面再进一步定义存取权限。

（4）授权

授权（Authorization）即定义用户的存取权限。授权有两种：系统特权和对象特权。

系统特权是由 DBA 授予某些数据库用户，只有得到系统特权，才能成为数据库用户。

对象特权包括对象许可和命令许可。对象特权既可以由 DBA 授予，也可以由数据对象的创建者授予，允许数据库用户对某些特定的数据对象进行某些指定操作。

数据库系统初始化时，系统中至少有一个具有 DBA 特权的用户，DBA 可以通过 GRANT 语句将系统特权或对象特权授予其他用户。对于已授权的用户可以通过 REVOKE 语句收回所授予的特权。

授权定义经过编译后以一张授权表的形式存放在数据字典中。授权表主要有用户标识、数据对象和操作类型 3 个属性。用户标识不但可以是用户个人，也可以是团体、程序和终端。在非关系系统中，存取控制的数据对象仅限于数据本身；而在关系系统中，存取控制的数据对象不仅有基本表、属性列等数据本身，还有内模式、外模式、模式等数据字典中的内容。

对于授权表，一个衡量授权机制的重要指标就是授权粒度，即可以定义的数据对象的范围。在关系数据库中，授权粒度包括关系、记录或属性。一般来说，授权粒度越细，授权子系统就越灵活，能够提供的安全性就越完善。

2．难点分析

第一小节提到的几种数据库安全措施，都是防止从数据库系统窃取保密数据，不能防止通过不正常渠道非法访问数据，例如，偷取存储数据的磁盘，或在通信线路上窃取数据，为了防止这些窃密活动，比较好的办法是对数据加密。数据加密是防止数据库中数据在存储和传输中失密的有效手段。

DBMS 是建立在操作系统之上的，安全的操作系统是数据库安全的前提。操作系统应能保证数据库中的数据必须由 DBMS 访问，而不允许用户越过 DBMS，直接通过操作系统访问。数据最后可以通过密码的形式存储到数据库中。

加密的基本思想是根据一定的算法将原始数据（Plain Text，术语为明文）加密成为不可直接识别的格式（Clipher Text，术语为密文），数据以密码的形式存储和传输。

加密方法有两种：一种是替换方法，该方法使用密钥（Encryption Key）将明文中的每一个字符转换为密文中的一个字符；另一种是转换方法，该方法将明文中的字符按不同的顺序重新排列。通常将这两种方法结合起来使用，就可以达到相当高的安全程度。

数据加密后，对于不知道解密算法的人，即使利用系统安全措施的漏洞非法访问数据，

也只能看到一些无法辨认的二进制代码。合法的用户检索数据时，首先提供密码钥匙，由系统进行译码后，才能得到可识别的数据。

3．典型例题

【例题 2-24】 某单位一信息项目的数据库平台采用了 Sybase SQL Server 作为后端平台。试写出以下安全相关的 SQL 语句。

【问题 1】 授予 zhang 在数据库 Student 上建表，建视图，建存储过程的权限。

【解析】 USE Student

GRANT CREAT TABLE，CREATE PROCEDURE，CREATE VIEW TO zhang

【问题 2】 授予 zhang 在数据库 Student 上对表 teachers 有 SELECT，reference 权限。

【解析】 USE Student

GRANT SELECT，REFERENCE ON teachers to zhang WITH GRANT OPTION

【问题 3】 授予 zhang 在数据库 Student 上对表 teachers 的 name，native 字段有 update 权限。

【解析】 USE Student

GRANT UPDATE ON teachers（name，native）to zhang

【例题 2-25】 阅读以下有关的叙述，回答问题 1 到问题 3。

数据的安全性是指保护数据以防止因不合法的使用而造成数据的泄密和破坏。这就要采取一定的安全保护措施。在数据库中，系统用检查口令等手段来检查用户身份，合法的用户才能进入数据库系统。当用户对数据库执行操作时，系统自动检查用户是否有权限执行这些操作。SQL Server 7 的身份验证过程中，有两种安全模式：NT 验证模式和混合式。

【问题 1】 在某一信息系统中，所有需要访问 SQL Server 的用户都有 Windows NT 的账户，而且 SQL Server 被安装在和用户账户域相同域中的一个 Standalone Server 上。所有的用户都使用 TCP/IP 和服务器连接。系统管理员应该使用哪种安全模式?

【解析】 NT 验证模式。

【问题2】在某一信息系统中，用户使用多种网络协议对网络进行访问，而且 SQL Server 所在的服务器是网络上唯一一台 Windows NT 服务器。应该使用哪种安全模式?

【解析】 混合安全模式。

【问题 3】 在某一信息系统中，系统需要让 Windows NT 的用户和 UNIX 工作站的用户都能够访问 SQL Server，并且希望在管理口令上花费尽可能少的精力，它应该使用哪种安全模式?

【解析】 混合安全模式。

【例题 2-26】 阅读以下有关的叙述，回答问题 1 和问题 2。

某单位一信息项目的数据库平台采用了 SQL Server 2000 作为后端平台，前端选择了 Visual Studio 作为开发工具，采用的 Client/Server 模式，项目采用小组开发的方式。

【问题 1】 创建了一张名为 np_file_come 的表，并且在 np_file_come 表上创建了一个

viewFileCome 的视图。并授予一个用户在 viewFileCome 上进行 SELECT 的权限，但该用户觉得视图中的字段太多，所以在上面又创建了一个叫 viewFileComeName 的视图。另外一个用户需要在工作中使用 viewFileComeName 中的数据，所以就为他分配了访问 viewFileComeName 的权限，请问要访问该用户能够得到 viewFileComeName 中的数据吗?

【解析】 SQL Server 2000 权限的检查是沿着所有权链，在主发生变化处进行的。第二个用户在第一个用户的视图上有权限，所以不用在第一个用户的表上检查权限。第二个用户在原始的视图上没有权限，所以他不能从中选取数据。

【问题 2】 原始创建的表 Weather，授予 public 组访问它的权限。随后，在这个表上创建了一个叫 viewLocalWeather 的视图，将它的访问权限授予了第一个用户。第一个用户在 viewLocalWeather 上又创建了一个叫 viewLocalTemperature 的视图，并为第二个用户授权访问这个视图。第二个用户能使用这个视图吗?

【解析】 不能。第二个用户在 viewLocalWeather 视图上没有 SELECT 的权限。所有权链的原理规定对象权限的检查是从上向下的，而且每当所有权发生变化时先检查视图。第二个用户想访问第一个用户的视图，而第一个用户要访问原始的视图，但是第二个用户无权访问原始的视图。

2.5　应用程序开发

应用程序开发是数据库应用系统实施的基本点，它包含了一系列复杂的过程，包括开发平台的选用，系统实施顺序，如何制定开发框架，多人、小组开发过程中如何控制源代码控制，多个版本迭代中版本控制的方法等。在程序的开发过程中，如何划分模块，使它们具有高内聚、低耦合的特点，软件工程学中有一套较为成熟的原则和方法。程序设计文档和设计评审又是保证软件质量的重要环节。

如图 2-9 所示是本节的知识框图。

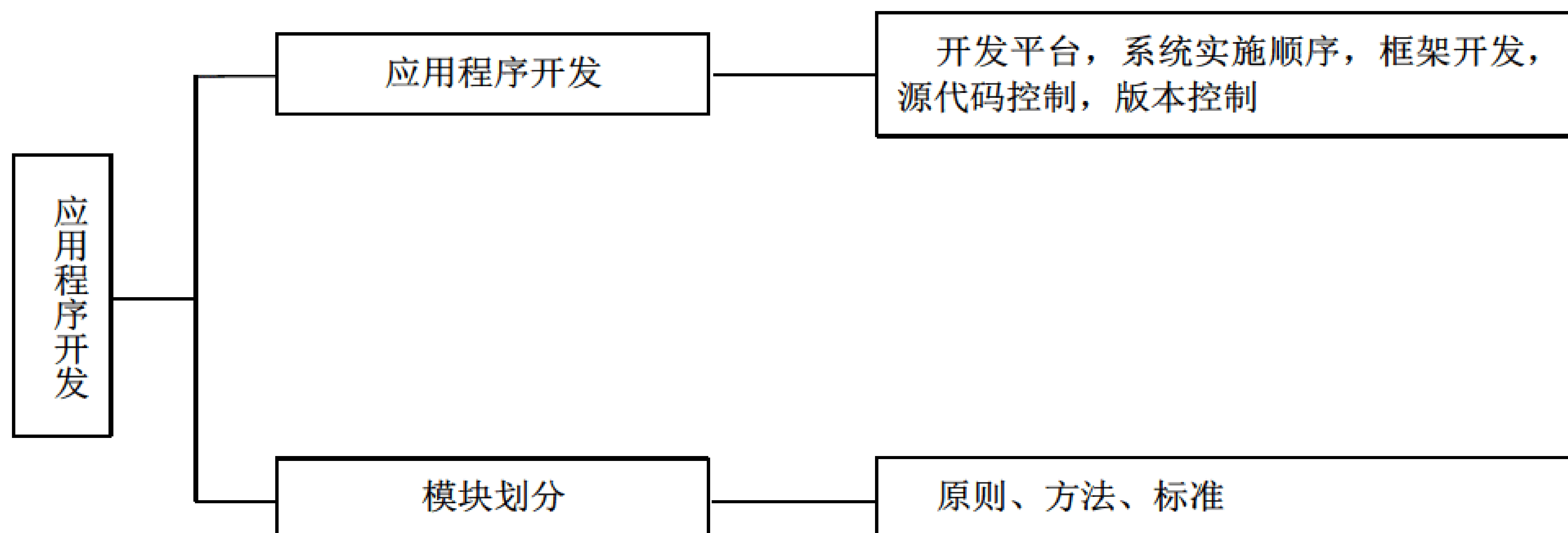


图 2-9　应用程序开发知识框图

2.5.1　应用程序开发

1．知识点提炼

（1）选择应用程序开发平台

应用程序的开发平台，即软件开发环境（Software Development Environment）是指支持软件产品开发的软件系统，它由软件工具集和环境集成机制构成。

用来辅助软件开发、运行、维护、管理、支持等过程中活动的软件称为软件工具，通常也称为 CASE（Computer Aided Software Engineering）即计算机辅助软件工程工具。软件工具集包括支持软件开发相关过程、活动、任务的软件工具，以对软件开发提供全面的支持。

环境集成机制为工具集成和软件开发、维护和管理提供统一的支持，它通常包括数据集成、控制集成和界面集成。通过环境集成机制，各工具用统一的数据接口规范存储或访问环境信息库；各工具采用统一的界面形式，保证各工具界面的一致性，同时为各工具或开发活动之间的通信、切换、调度和协同工作提供支持。

选择合适的应用程序开发平台进行软件开发，不仅可以使用软件开发环境中提供的各种工具，而且在环境信息库的支持下，一个工具所产生的结果信息可以被其他工具利用，使得软件开发的各项活动得到连续的支持，从而提高了软件的开发效率。

一般来说，软件开发环境具有如下特征。

① 开发环境的服务是集成的。软件开发环境应支持多种集成机制，如平台集成、数据集成、界面集成、控制集成和过程集成等。

② 开发环境应支持小组工作方式，并为其提供配置管理。

③ 开发环境的服务可用于支持各种软件开发活动，包括分析、设计、编程、测试、调试、文档等。

衡量一个应用程序开发平台的优劣，可以参考如下标准。

① 功能；

② 稳健性；

③ 易用性；

④ 硬件要求；

⑤ 服务和支持。

（2）系统实施顺序

经过详细设计后，便进入系统的实施阶段。系统实施是指将系统设计阶段的结果在计算机上实现，将原来纸面上的、类似于设计图的系统方案转换成可执行的应用软件系统。系统实施的主要任务和顺序如下所述。

① 硬件准备。按照总体设计方案的要求和可行性报告中财力资源的分析，购置和安装计算机网络系统。硬件准备包括计算机主机、输入/输出设备、存储设备、辅助设备（稳

压电源、空调设备等）、通信设备等。

② 软件准备。软件准备包括系统软件、数据库管理系统以及一些应用程序。这些软件有些需要购买，有些需要组织人员编写。

③ 建立数据库系统。

④ 应用程序设计和实现。编写程序是系统实施阶段的重要任务之一。

⑤ 数据准备。数据的收集、整理、录入是一项繁重的工作。一般来说，确定数据库模型之后，就应该进行数据的整理、录入。这样既可以分散工作量，又可以为系统调试提供真实的数据。

⑥ 系统测试。

⑦ 用户培训。

⑧ 系统转换和试运行。

（3）框架开发

框架开发是信息系统开发过程中另一个重要阶段，其主要目标是为系统制定蓝图，为系统搭好框架。在各种技术和实施方法中，合理使用各种资源，勾画出系统的详细实施方案。

框架开发的主要内容包括系统的总体结构设计、代码设计、输出设计、输入设计、处理过程设计、数据存储设计、用户界面设计和安全控制设计等。框架开发大致可以分为以下两个步骤。

首先，把总任务分解为多个基本的、具体的任务。将系统合理划分为模块，确定每个模块的功能、模块之间的调用关系以及模块间信息的传递。

其次，为各个具体模块选择适当的技术手段和处理方法，内容包括代码设计、输出设计、输入设计、处理过程设计、数据存储设计、用户界面设计、安全控制设计。

为了保证框架开发的顺利完成以及之后的开发能够顺利地进行，框架开发应该遵循如下原则。

① 系统的分解与模块协调相统一。整个系统是一个庞大的整体，具有整体目的和功能，为了实现整体的功能，必须把它分解成多个子系统分别处理，相互联系的子系统共同作用实现整体的功能。在对整个系统进行分解时，要根据系统的总体要求协调各个子系统的关系。

② 首先把握系统总的功能，然后将总体功能逐层分解成多个子功能。即先确定上层模块的功能，再细分下层模块的功能。

③ 处理好上下层模块之间的关系。上层模块只规定下层模块做什么和所属模块间的协调关系，但不规定怎么做，以保证各模块的相对独立性和内部结构的合理性，使得模块与模块之间层次分明，便于实施和维护。

④ 各个模块必须功能明确、接口明确。模块之间的耦合尽可能小，模块内部组合要尽可能紧凑。

⑤ 模块的大小要合理。如果模块过大，说明系统分解得不充分，模块内部可能包含了若干部分的功能，所以应该进一步把原有的模块分解成若干功能尽可能单一的模块。如果模块过小，则有可能降低模块的独立性，造成系统接口的复杂性。

（4）基础小组的程序开发

许多大型软件都是由小组开发的。小组必须对项目做出规划，跟踪项目的进度，并协调小组成员的工作。同时，小组成员还必须对工作目标取得共识，有一个一致的工作流程，并且可以自由地、经常地互通信息。

要赶上紧迫的进度表并且生产出高质量的产品，训练有素的小组协同工作是其中的关键。然而，训练有素的小组协同工作需要大量的经验和一整套专门的技能与方法。读者应该了解简单的小组软件开发过程，小组研究周期的详细内容，如何在小组中制定开发策略，进行开发设计，定义需求，进行小组设计，实现产品，进行集成和系统测试和后期维护；小组软件开发的步骤；小组中的角色：小组领导者、开发经理、计划经理、质量/生产经理，以及技术支持经理；和小组软件开发需要注意的一些原则等。

（5）源代码控制

大型的软件开发常常是由多个程序员进行的，随着越来越多的软件开发人员参与一个项目的开发，管理所有开发人员所做的工作就变得更加困难。小组软件开发必须解决以下问题。

① 确保多个开发人员不在同一时间修改同一个代码和对象；

② 防止软件开发人员改写其他人的工作；

③ 跟踪软件的版本；

④ 将各个项目文件集中起来。

上面提到，当两个开发人员同时修改同一个代码时，有个人的工作就会丢失。一般来说，完成修改的第一个程序员将会丢失他所做的修改工作，因为第二个程序员改写了第一个程序员所做的修改。要想确保项目开发过程中开发人员不会互相改写对方的工作，这是非常困难的。另外，多个程序员还必须注意不要为了解决同一个问题而浪费时间。

在小组开发环境中，版本的跟踪也是需要解决的问题。由于对那么多的源文件做了那么多的修改，因此很难得到项目在特定状态下的瞬态图。当开发人员修改不同文件或者修改相同文件时，跟踪这些修改会变得越来越复杂。

（6）版本控制

软件的开发、维护和升级，往往是由多个开发人员共同协作的一个过程。不同人对同一个软件的不同部分同时做着修改，这种行为有时会出现彼此交叉的情况，常常出现代码不一致的现象。如果将这种一致性错误的纠正延迟到测试阶段，则会增加调试的难度，从而降低开发效率。版本控制由此而生。

版本控制的作用是跟踪记录整个软件的开发过程，包括软件本身和相关文档，辅助协调和管理软件开发团队。进行版本控制，所带来的好处有：可以标识不同阶段的软件及相

关文档，进行差别分析；对软件进行可撤销的修改；便于汇总不同人员所做的修改。

对于一个采用版本控制进行软件开发的多人开发团队而言，其一般的开发方式是：采用 Client/Server 的结构，分别在客户端和服务器上安装版本控制工具的客户端和服务器版本，软件放在服务器上为开发人员所共享，开发人员在客户端从服务器上将软件的相关部分下载到本地，进行修改，最后将改动结果提交到服务器上。

衡量版本控制的效果有两大标准：效率和质量。如果版本控制最终使软件开发效率得到提高、使软件质量（在此，软件质量主要有软件的一致性和冗余程度两大指标。）得到提升，则这样的版本控制是成功的。

2．难点分析

前台应用程序开发工具的选用是一个难点。

Visual Basic（VB），它是以 Basic 语言作为其基本语言的一种可视化编程工具。VB 作为一种较早出现的开发程序，其优点是容易学习、开发效率较高、具有完善的帮助系统等。它对组件技术的支持是基于 COM 和 Active X，在组件技术不断完善发展的今天，它显出了落后性。VB 在进行系统底层开发的时候也是相对复杂的，调用 API 函数须声明，调用不方便，不能进行 DDK 编程，不能嵌套汇编。VB 面向对象的特性差。其网络功能和数据库功能也没有非常突出的表现，数据连接使用 DAO、ADO、RDO，数据表现时使用 DBGrid，与数据库相关的数据表现控件只有此一种，只能表现简单表格数据，表现手段单一。SQL 语言的支持方式将一句 SQL 串绑定到一个命令对象中，结果返回到 ResultSet 对象中。VB 的移植性较差。

Power Builder，是开发 MIS 系统和各类数据库跨平台的首选，使用简单、容易学习、容易掌握，在代码执行效率上也有相当出色的表现。PB 是一种真正的 4GL 语言（第 4 代语言），可随意直接嵌套 SQL 语句，返回值被赋值到语句的变量中，支持语句级游标、存储过程和数据库函数，是一种类似 SQLJ 的规范，数据访问中具有无可比拟的灵活性。但是它在系统底层开发时，同样有调用 API 函数须声明，调用不方便，不能进行 DDK 编程，不能嵌套汇编等缺点。在网络开发方面提供了较多动态生成 Web 页面的用户对象和服务以及系统对象，非常适合编写服务端动态 Web 应用，有利于商业逻辑的封装；但是用于网络通信的支持不足；静态页面定制支持有限，使得 PB 在网络方面的应用也不能非常广泛。面向对象特性也不是太好；数据连接使用 Transaction、DwControl；可绑定任何 SQL 语句和存储过程；数据访问具有很好的灵活性；数据表现时使用 DataWindow 对象。SQL 语言的支持方式为：可随意直接嵌套 SQL 语句。返回值被赋值到语句的变量中，支持语句级游标、存储过程和数据库函数，是一种类似 SQLJ 的规范。移植性方面支持 Windows 家族、Solaris、Macintosh Windows 家族、Macintosh 等。

Delphi 是基于 VCL 库的可视化开发工具，它们在组件技术的支持、数据库支持、系统底层开发支持、网络开发支持、面向对象特性等各方面都有相当不错的表现，并且学习使用较为容易，充分体现了所见即所得的可视化开发方法，开发效率高，代码执行效率高。

但是帮助系统在众多的编程工具中是属于比较差的，特别是对于中文帮助的支持很差。另外，Delphi 的语言 Object Pascal 应用不够广泛。数据连接使用包括 DataSource、Table、Query、Midas、ADO 在内的 20 多个组件和类完成数据访问，数据表现时使用包括 DBGrid、DBNavigator、DBEdit、DBLookupListBox 在内的 15 个数据感知组件，DecisionCube、DecisionQuery 在内的 6 个数据仓库组件和包括 QRChart、QRExpr 在内的 20 多个报表组建，可灵活表现数据。SQL 语言的支持方式为：数据库组件或类完成 SQL 语句串的执行和提交。移植性方面支持 Windows 家族、Linux 等。

C++ Builder 是基于 VCL 库的可视化开发工具，具有和 Delphi 类似的优点。由于 C++ Builder 库是基于 Object Pascal（面向对象 Pascal），使得 C++ Builder 在程序的调试执行上落后于其他编程工具。VCL 库本身是 Object Pascal，兼容性不好。对数据库的支持和 Delphi 类似。

Visual C++是基于 MFC 库的可视化开发工具，从总体上说它是一个功能强大但是难于使用的一种工具。它在网络开发和多媒体开发上都具有不俗的表现，帮助系统也做得非常不错。但是为了兼容 C 的程序，在面向对象特性上却不是很直观。在组件支持方面，支持 COM、ActiveX 和 CORBA，但是没有任何 IDE 支持，有所有 C 编译器的功能，需要 CORBA 中间件支持。它最大的问题是开发效率不高。在数据库方面，MFC 提供不少支持数据库的类库可供使用，但不直观，开发效率很低。Visual C++在移植性方面也较差。

基于 Java 编程工具，目前比较出名的是 Borland 出的 JBuilder 和 IBM 的 Eclipse，两种工具都有一定数量的使用人群。JBuilder 继承了 C++ Builder/Delphi 的特点，在可视化上做得非常不错，使用简便。由于 Java 本身语言的特点使得在网络开发中具有高人一等的表现，而且面向对象特性高，支持的组件技术也非常多，跨平台的特性也使得它在现在和未来的开发中占据越来越重要的地位。但是在系统底层开发和多媒体开发中却表现得差强人意。数据连接和数据表现时使用 Java 和 JDBC API。另外，不同的 IDE 具有不同的组件，SQL 语言的支持方式为：SQLJ 和 Java JDBC API。移植性方面很好。

3．典型例题

【例题 2-27】 在应用程序设计时，常常使用一些经典的模式来提高应用的健壮性、灵活性和可扩展性。下面的 Java 程序段中使用了什么样的设计模式？

```
public class A {
        private static A singleton;
        private A(){
        }
        public static A getInstance(){
                if(singleton==null)singleton = new A();
                return singleton;
        }
```

```
}
public class B {
        A a=A.getinstance();
}
```

【解析】 单件模式。

【例题 2-28】 阅读以下有关的叙述，回答问题 1 和问题 2。

每一个软件项目，无论是工程类项目，还是产品类项目，都必须经历需求分析、系统设计、编码实现、集成测试、部署、交付、维护和支持的过程。在这个过程中，将生成各种各样不同的文件，包括文档、源程序、可执行代码、支持库。更可怕的是：频繁的出现变更是不可避免的，因此面向如此庞大且不断变动的信息集，如何使其有序并且高效地存放、查找和利用就成了一个突出问题。

某公司的一些开发人员主要从事 Windows 和 Linux 平台下的软件开发，采用的工具包括 Visual Studio 系列、GCC 等。为了能够加强版本控制与配置管理工作，决定引入一些自动化配置管理工具。

经过慎重的选择，采用了两步走的方法：首先采用了 Visual Studio 软件包中的 VSS（Visual Source Safe）作为配置管理工具。由于 VSS 安装、配置、操作都十分简单，上手容易，这样在执行配置管理的过程中，工具的培训没有带来太大的阻力，大家可以集中精力理解配置管理。这样很快就在团队中形成了版本控制、配置管理的氛围与习惯。然后构建了 CVS 服务器，作为整个开发组织的配置管理工具；CVS 能够有效地支援 Windows、Linux 两个平台上的应用开发，其性能优秀，而且免费，另外，它对于兼职人员的配置管理十分有效。采用 CVS 至今，效果明显，除了功能、使用上有些不方便之处外，其他功能都能达到预期的目的。

【问题 1】 除了以上案例中提到的版本控制工具，在日常开发中，还有哪些比较成熟的版本控制工具？

【解析】 比较早期的版本控制工具 CCC、SCCS 和 RCS，重量级软件配置工具 Rational 公司的 Rational ClearCase，工作小组级的 Merant PVCS，Microsoft 公司的入门级版本控制工具 Visual Source Safe，国内北大青鸟的 CASE 工具等。

【问题 2】 根据自己的项目经验，列举选用版本控制工具时的注意点。

【解析】选用版本控制工具时，首先应考虑功能是否符合实际需求，是否符合团队特点，例如工具对并行开发支持、对异地开发支持、对跨平台开发支持和与开发工具的集成性等；然后还应考虑的问题有：易用性和运行性能、工具的安全性、费用是否可以接受、售后服务等其他因素。

【例题 2-29】 阅读以下关于软件工程管理方面的叙述，回答问题 1 和问题 2。

某大型企业集团的信息工程部的软件工程师从事着企业内外的软件开发与维护工作，该集团分布地域广阔，集团内采用了多种操作系统平台和多类开发环境。总工程师在总结

近 3 年来的软件开发工作时，发现有 15%左右的软件开发项目未能完成而被迫取消，其余 85%的项目中大多也不能完全实现预定的目标。比如平均每个项目的实际完成成本超过预算 152%，平均完成时间超过预期的 216%。这些数据表明信息工程部未能对开发成本和开发进度实施有效的控制。

信息工程部召开了 3 次小结与分析会议，在会上集中讨论了软件工程管理有关的问题，在归纳的意见中出现了以下的一些内容。

① 软件开发已经逐渐成为一类工业化的生产过程，必须尽可能对其中的所有环节进行有效的管理与控制。

② 软件工程管理与其他工程管理相比，主要的困难包括：软件产品的不可见性（难以把握开发进展与质量要求等）、软件开发过程的非标准化和许多软件项目开发的“一次性”（缺少可借鉴的经验）等。

③ 软件开发面对着进度、成本、功能和性能 4 方面的主要约束，即要求在预定的期限内，使用规定的有限资源，满足不断增长的软件功能与性能需求。从这个角度来看，必须强化软件工程的管理。

④ Client/Server 与 Browser/Server 模式等已成为当前软件体系结构的主流。在日益复杂的分布式开发环境下，进行跨平台的团队开发，实现代码共享相当困难，软件工程管理在其中可以发挥其重大作用。

⑤ 尽可能采用更加先进的操作系统、工作平台和开发工具，可以极大地提高软件开发效率，从根本上防止和解决在总结中所发现的软件开发问题。

⑥ 面向对象（OO）技术的使用，可以保证在跨平台的复杂环境下团队开发的需要，从而可从根本上免除软件工程管理上的烦恼，也能从根本上缓解所发现的软件开发问题。

⑦ 软件开发项目未能完成而被迫取消，究其根本原因无非是两大类：一是用户需求变更过于频繁或要求过高，另一是软件开发人员的素质不能适应项目要求。

⑧ 一个大中型应用系统的开发可能包含有成百上千个软件模块以及数以百万的代码行，任何一个编程人员不可能了解和追踪该应用系统的每一个片断，因此使软件代码具有可管理性和可审核性将是软件工程管理中的一项重要任务。

⑨ 软件在投入使用后的维护与支持工作极为重要，软件工程管理在这方面也可以发挥其重要作用。

⑩ 本集团的长远目标是需要建立一类软件开发管理体系，能有效地辅助软件开发全过程中对相关信息的收集和管理，这类体系应是可重复使用并可适用于各类软件开发项目，使软件资源在软件的生命周期中保持完整。

对外服务的软件开发机构应当努力取得 ISO 9000 质量认证，并根据 CMM（能力成熟度模型）标准来改进自己的开发过程。在这些质量标准实现的过程中，软件工程管理起着重要的作用。

【问题 1】 工程师指出在上述的 11 条意见中，有 3 条的提法是不够恰当或是不够全面

的，请指出其序号，并各在 50 字以内说明理由。

【解析】 不够恰当或是不够全面的 3 条意见如下所示。

第 5 条，从目前情况看，先进工具与平台至多仅包含部分软件工程管理辅助，还不可能取代软件工程管理全部内容。

第 6 条，跨平台复杂环境多重结构开发含有许多需要加以管理的对象类型，多样化的团体开发也应加强管理。

第 7 条，更深层看，项目失败主要是缺乏有效的软件工程管理机制所引起的。

【问题 2】 信息工程部在详细分析本集团的软件工程管理方面存在的各类问题时，发现在不少软件项目实施过程中，软件项目计划这一环节较为薄弱，尤其是对软件项目资源估算问题上有一些偏差。软件开发所需的资源估算包括人力资源、软件资源和硬件与系统平台资源 3 大部分，目前该集团的一个主要矛盾是在软件开发所需要的“软件资源”的估算与计划上。

请在 100 字以内以提纲方式说明软件开发中所需的软件资源主要包括有哪些具体的内容？（只须列出相应的名称即可）

【解析】 需计划的软件资源有两大类：软件工具集（CASE），如业务系统计划工具集，项目管理工具集，支持工具（如文档生成），分析与设计工具，编程工具，组装与测试工具，原型化与模型工具，维护工具，框架工具等；（只需要回答出有代表性的一部分软件工具集即可，也可只列出某个集成的工具集，如 IPSE 等）可复用软件与软件构件库。

2.5.2　模块划分

1．知识点提炼

模块是组成系统的基本单位，它的特点是可以组合、分解和更换。系统中任何一个处理功能都可以看成是一个模块。

一个模块应该具备如下 4 个要素。

① 输入和输出，模块的输入来源和输出去向都是同一个调用者，即一个模块从调用者那里取得输入，进行加工后再把输出返回给调用者。

② 处理功能，指模块把输入转换成输出所做的工作。

③ 内部数据，指仅供该模块本身引用的数据。

④ 程序代码，指用来实现模块功能的程序。

为了便于之后的系统开发和系统运行，模块划分要遵循如下的原则。

① 所划分的模块其内部的凝聚性要强，模块之间的联系要少，即模块具有较强的独立性。

② 各个模块之间数据的依赖性应该尽量小。模块之间的连接只能存在上下级之间的调用关系，不能有同级之间的横向联系。整个系统呈树状结构，不允许网状结构或交叉调用关系出现。

③ 模块划分的结果应使数据冗余较小。

④ 所有模块都必须严格地分类编码并建立归档文件。

⑤ 模块的划分应便于系统分阶段实现，并使各类资源可以得到充分利用。

2．难点分析

模块化是软件设计的一个基本准则。

抽象就是抽出事务的本质特性而暂时不考虑它们的细节，模块是按照不同的抽象级别安排的，高层抽象模块向读者隐藏了功能实现的细节，这就是信息隐蔽，模块之间相互隐藏自身的实现细节对一个好的设计来说是至关重要的。

耦合是对不同模块之间相互依赖程度的度量。紧密耦合是指两个模块之间存在着很强的依赖关系；松散耦合是指两个模块之间存在一些依赖关系，但它们之间的连接比较弱；无耦合是指模块之间根本没有任何连接。

耦合的强度依赖于以下 4 个因素：一个模块对另一个模块的引用；一个模块向另一个模块传递的数据量；一个模块施加到另一个模块的控制的数量；模块之间接口的复杂程度。

从强到弱的几种常见的耦合类型为：内容耦合，一个模块直接修改或操作另一个模块的数据；公共耦合，两个以上的模块共同引用一个全局数据项；控制耦合，一个模块在界面上传递一个信号控制另一个模块，接收信号的模块的动作根据信号值进行调整，称为控制耦合；标记耦合，若两个模块至少有一个通过界面传递的公共参数包含内部结构；数据耦合，模块间通过参数传递基本类型的数据，数据耦合是最简单的耦合形式，系统中至少必须存在这种类型的耦合。

内聚度量的是一个模块内部各成分之间相互关联的强度，如果一个模块的所有成分都直接参与并且对于完成同一功能来说都是最基本的，则该模块是高内聚的。

从低到高的几种常见的内聚类型为：偶然内聚，一个模块的各个成分之间毫无关系；逻辑内聚，几个逻辑上相关的功能被放在同一模块中；时间内聚，一个模块完成的功能必须在同一时间内执行，但这些功能只是因为时间因素关联在一起；过程内聚，一个模块内部的处理成分是相关的，而且这些处理必须以特定的次序执行；通信内聚，一个模块的所有成分都操作同一数据集或生成同一数据集；顺序内聚，一个模块的各个成分和同一个功能密切相关，而且一个成分的输出作为另一个的成分；功能内聚，最理想的内聚是功能内聚，模块的所有成分对于完成单一的功能都是基本的。

内聚和耦合是密切相关的，在进行软件设计时，应力争做到强内聚、弱耦合。

3．典型例题

【例题 2-30】 阅读以下关于办公室自动化系统工作流分析方面的叙述，回答问题 1 到问题 3。

某市政府部门在网络环境下以若干个数据库为中心，已形成了一个初步的 OA 系统，并且正常运行了几年。该 OA 系统能处理常规的公文收发、归档处理、事务管理和业务信息汇总等基本功能，办公人员也已能熟练地使用计算机从事文字处理、电子报表、日程管

理、电子邮件、演示材料制作以及数据库的建立与使用等各类工作。

信息处在小结该政府部门的办公自动化OA工作时提出了以下意见。

① 目前的OA应用大多还处于局部个人工作的状态。

② OA对提高政府办事效率的作用尚不够显著。

③ 对有关领导人员进行决策分析的支持也很不够。

为此，组织了一个小组调查采用工作流方法改造OA系统的可能性，调查结果如下所述。

① 该政府部门办公室系统中的工作流是一类有连贯性的工作过程，一般包括有若干个明确定义好的子任务及其相互之间的顺序与连接关系。

② 可以认为，一个工作流包括工作流程的启动与终止条件、有关子任务的详细描述与实施、子任务的时序与调度、相关办公人员的权限指定与素质要求、对应的应用程序与数据资源等许多内容。

③ 在计算机网络系统中有可能实现基于工作流的方案（比如逻辑上的工作流），允许提高办公室中许多关键性工作任务的组织与实现效率，也可能为提高领导人员的决策与调度能力提供有效的支持。

④ 在该政府部门的OA应用中，需要有一个工作流的管理系统与机制，用来实现对工作流的定义、提取、执行、记录与监控等活动。

【问题1】 工作小组为了比较原来的OA系统与将来可能实现的基于工作流的系统，把原来基于功能模块的OA系统简要的表示为：

$SF=\{FM_1, FM_2, \cdots, FM_k\}$；

$FM_i=\{T_{i1}, T_{i2}, \cdots, T_{is}\}$；

$TFM=\{T_{ij}\}$。

其中：SF为基于功能模块的系统，FM_i为功能模块，T_{ij}为功能模块FM_i中的子任务，TFM为原系统中全体子任务的集合。

如果在分解子任务时，尽可能地做到两个子系统中相应子任务的基本功能大体相同，最终发现子任务的集合TWF相同于TFM。这时，工作组认为基于工作流的系统SW的功能包含了基于功能模块的系统SF，并且SW还包含有SF中所没能提供的许多信息。

请在100字以内，简要列出在SW系统中可能会增加哪些方面的信息？

【解析】 各类任务、人员（或代理）的协作信息、时序信息以及工作流有关的控制与监视信息。

【问题2】 工作小组认为在基于工作流的OA应用系统中，核心部分是要建立起一个OA工作流管理环境，如图2-10所示。这个工作流管理环境包括：管理机构；机构、人员、代理的组织与授权方面的信息及其管理；子任务管理；工作流的定义及其管理；流程相应数据的控制与管理；监控与异常处理；工作流引擎。

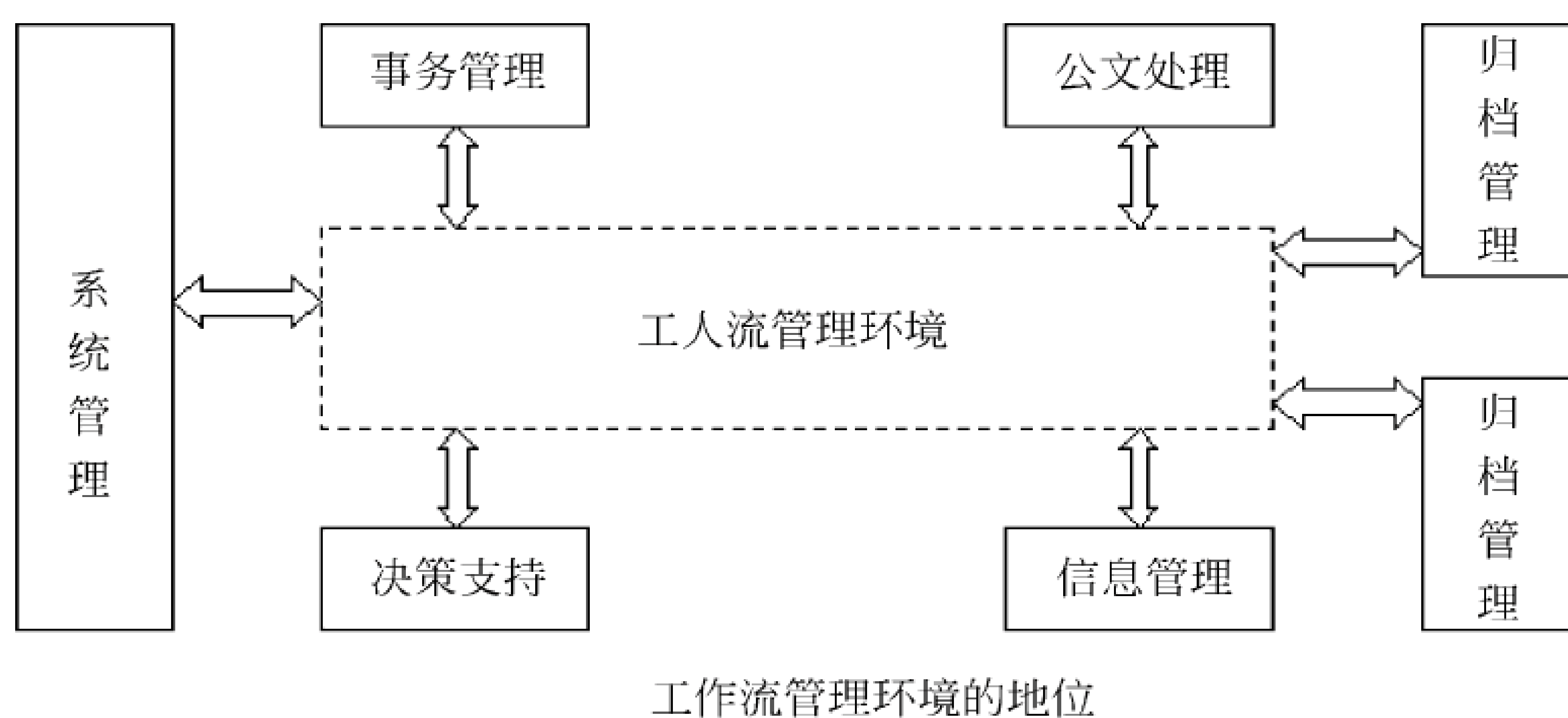


图 2-10　工作流管理环境的地位

请分别在 50 字以内回答下列问题。

① 工作流管理环境的根本目标是什么？

② 工作流引擎的主要功能是什么？

【解析】 答案如下。

① 工作流管理环境的根本目标是解决办公环境的协作。

② 工作流引擎的主要功能是根据工作流的定义，创建与控制执行工作流，调度、监控与管理类功能。

【问题 3】 根据自己开发 OA 或 MIS 系统工作的实践，请为本题的网络列举出一类比较适合开发基于工作流的 OA 系统的流行平台及其相关的软件或工具，包括对关系数据库的存取要求（在 100 字以内简要列举）。

【解析】 允许答案对目前流行的 Lotus Domino、MS Exchange 或 Group Wise 等选一种即可，比如采用 Lotus Domino Server 平台，使用 Notes 群件技术，（Notes R5 之前）ODBC 访问简单数据库，也可进一步用 Lotus Script；采用 MS Exchange Server 平台，使用 Visual Studio 和 Outlook，数据库访问有关 ADO 与 ODBC，也可用 OWA（outlook web access）、IIS、Web 网 ASP 支持等。

【例题 2-31】 阅读以下关于企业成本管理方面的叙述，回答问题 1 到问题 3。

某汽车配件生产企业希望采用供应链管理模式加强对本企业的生产作业成本进行管理。企业信息科负责计划开发一个新的企业成本结算与管理系统，如图 2-11 所示。

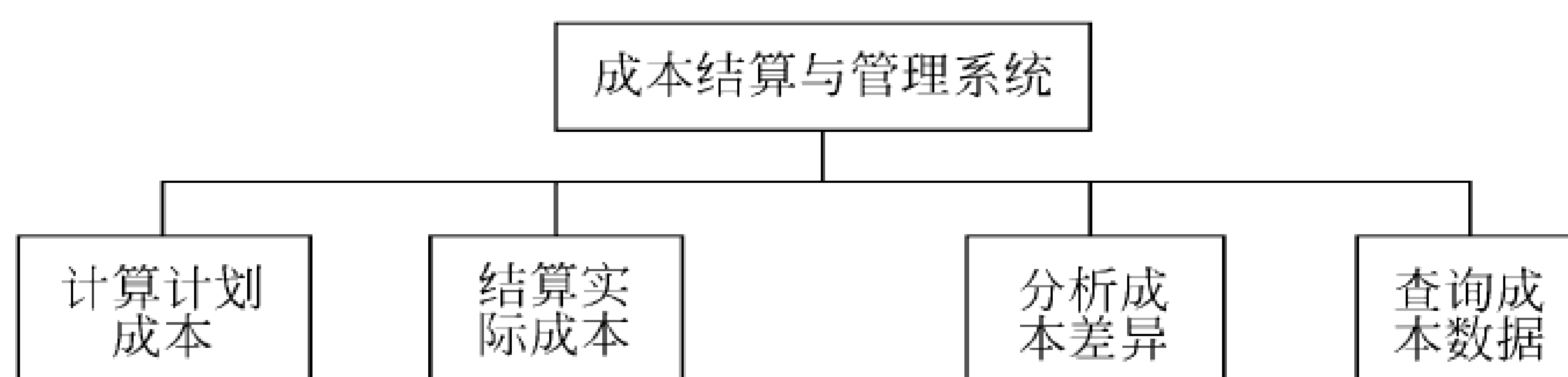


图 2-11　成本结算与管理系统

企业负责人召集了财务人员、管理人员和信息科技术人员一起拟定了该系统的实施要点，包括以下内容。

① 依据市场上材料价格的动态实际变化情况，把产品成本结算划分为“计划价格成本”、“平均价格成本”和“最新价格成本”3 类价格体系，分别进行计划成本计算和实际成本结算，从而使企业能更科学地实现产品定价策略，便于有效地打开市场和占领市场。

② 采用基于生产活动的作业成本核算方法，使成本的结算具体落实到车间、工序和零件（材料），即能使成本结算的过程与生产活动作业过程并行地对照开展。

③ 能依据产品的实际成本和计划成本进行“成本差异”分析，分析的结果也可落实到车间、工序和零件（材料），同时提供成本差异的“原因分析”和“结构分析”，有助于企业领导了解成本的消耗过程，对成本控制进行分析决策，在管理上实施相应的改进措施。

④ 按现行企业成本会计结算的周期，各类成本结算按月进行，即每月运行一次。每年再进行年度成本结转，通常会把上一年成本余额结转到下一年度作为该年年初成本。在系统数据库中，年初成本可用“零”月成本表示，到 1 月份结转时把零月数据结转进去。即企业采用单位成本和增量成本为基础的成本结算方法，有利于使成本数据与生产作业活动更加有机地联系起来，适应于企业现代化成本管理的需要。

⑤ 计划采用三层 Client/Server 结构，建立起性能良好的网络应用服务器和数据库服务器，使用信息科已十分熟悉的前端开发工具有效地加速系统的开发工作。

如图 2-12～图 2-14 所示是本系统中大致的工作流程。

以“结算实际成本”为例，基本数据来源于企业基础数据子系统，库存子系统和车间子系统。当输入生产数据时，系统应把车间子系统中的生产数据送入相应的数据库中，发送完毕后可浏览相应的数据，然后运行。

【问题 1】 在图 2-12 和图 2-13 中都有“校验数据”这一流程，请在 100 字以内简要说明该框的主要作用。

【解析】 检验有关数据库，保证所需要的数据齐全、规格正确（一旦有错可与相关部门联系纠正后继续运行）。

【问题 2】 你认为在这个成本结算与管理系统中“查询成本数据”可以提供哪些需要的查询功能？请在 100 字以内简要列出。

【解析】 按产品、按年度、按月、按车间、按工序、按零件（材料）查询。

【问题 3】 选定年份、月份和一个要分析的产品，请提供成本差异原因分析的一种理想的层次结构分析方法（在 100 字以内说明）。

【解析】 按产品结构树自然地逐层展开，每层应显示出成本差异最大的零件编号和成本差异值。也可分解成本差异为材料费差异（又可分为材料价格差异和材料用量差异）和工时费差异（单件工时差异）。

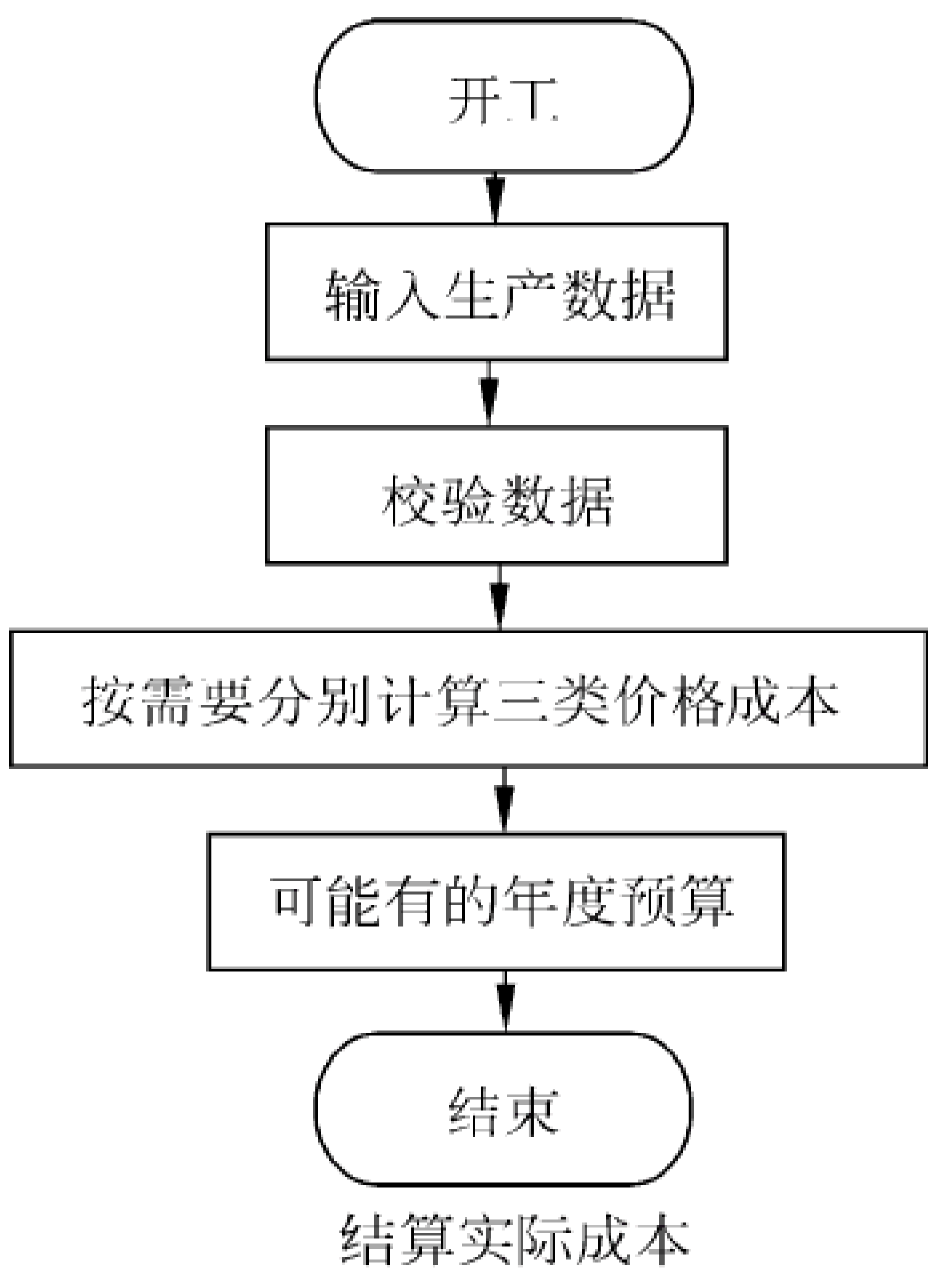


图 2-12　结算实际成本

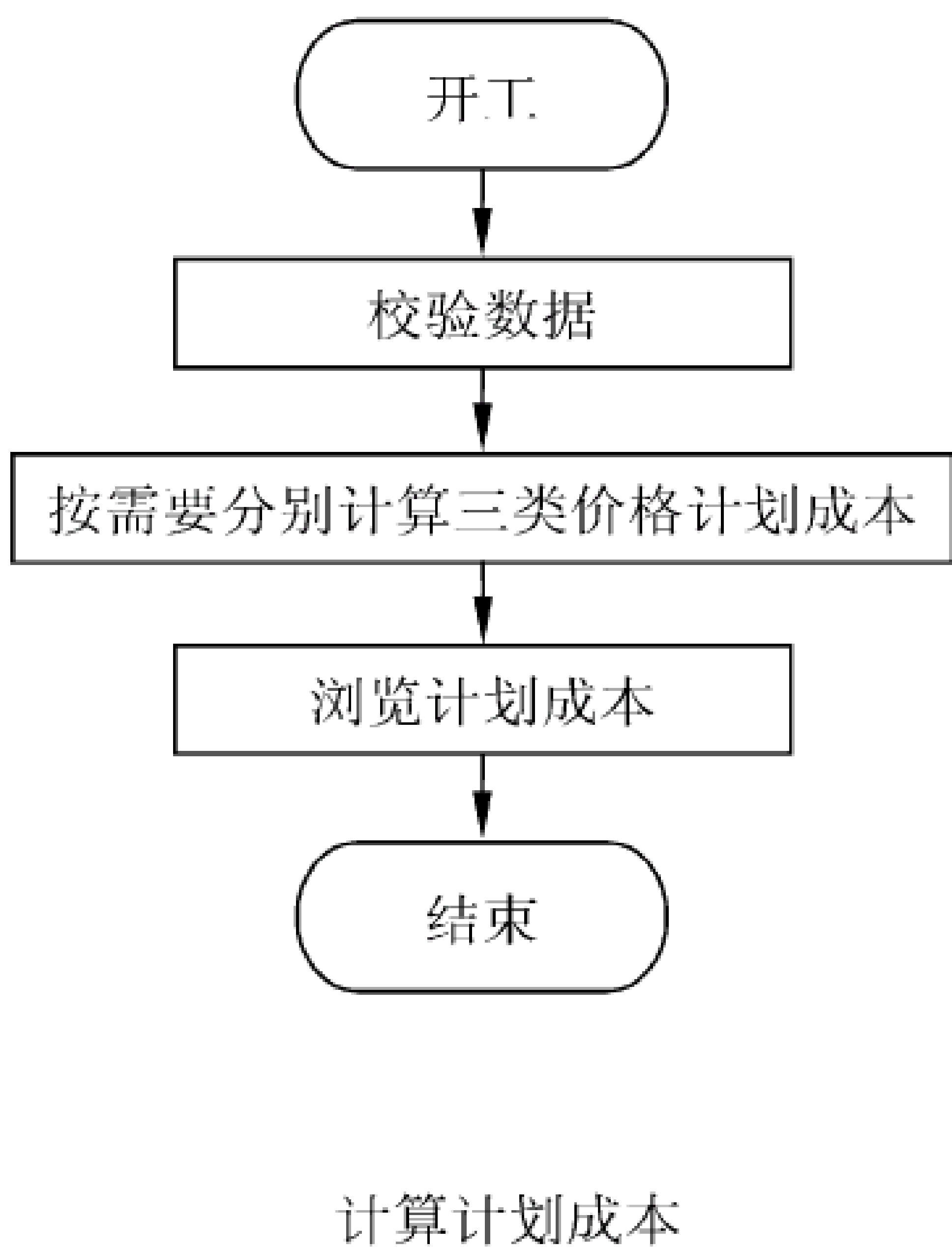


图 2-13　计算计划成本

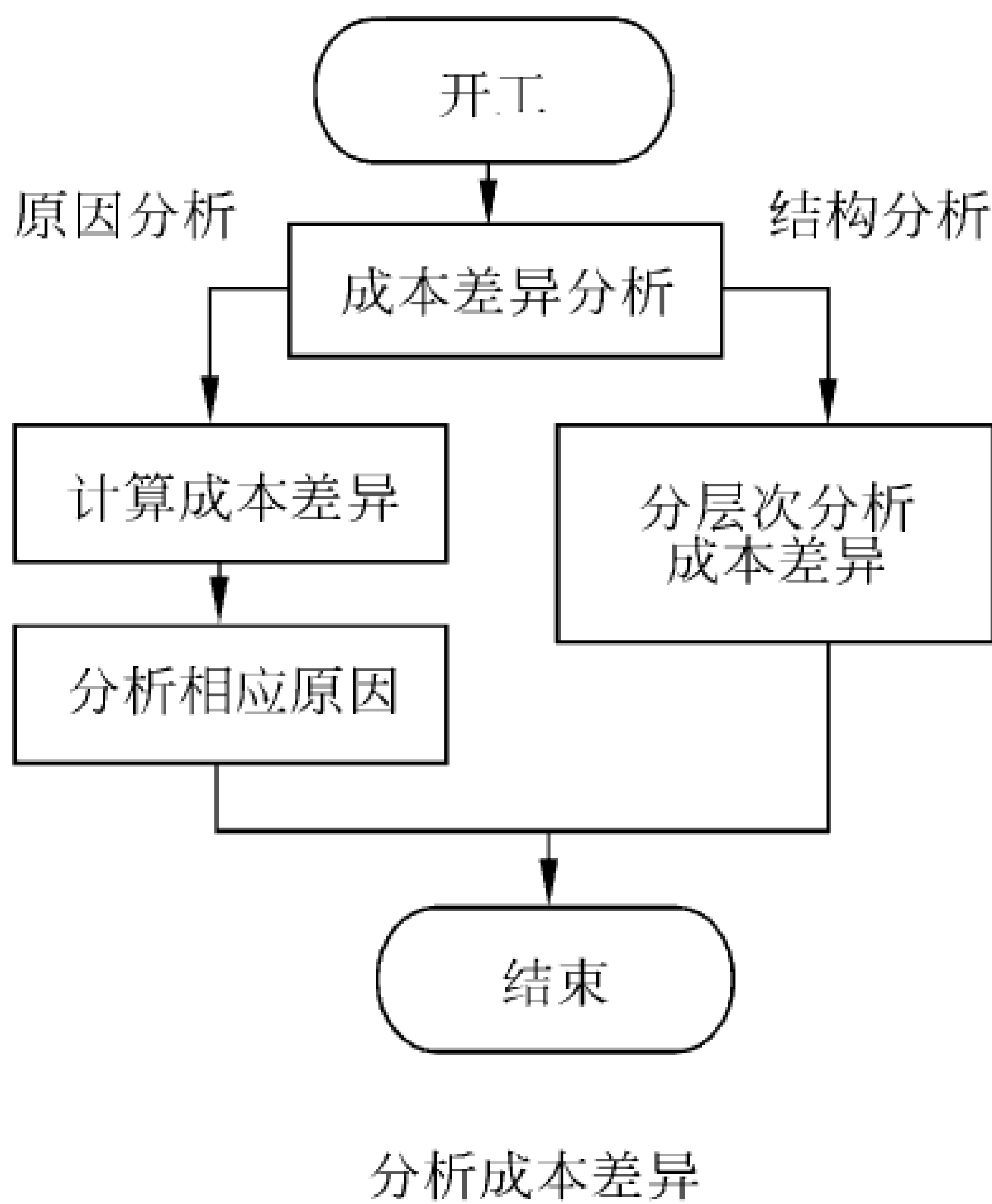


图 2-14　分析成本差异

2.6　编写应用系统设计文档

数据库应用系统开发过程中，文档是项目管理和软件质量保证中的一个重要方面。按照不同的开发模型和不同的开发阶段，文档编写的目的和内容也不一样。系统规划和系统分析阶段的文档主要包括系统可行性研究报告、系统总体规划报告、系统开发合同、系统方案说明书等；系统实施阶段的文档主要有系统开发计划、系统开发月报以及系统开发总结报告等项目管理文件；系统测试阶段需要用到的文档主要有系统方案说明书、系统开发合同、系统设计说明书、测试计划、系统测试报告；系统运行阶段涉及的文档主要是指用户手册和操作指南。

如图 2-15 所示是本节的知识框图。

1．知识点提炼

下面主要讲述系统配置说明、构件划分图、构件间的接口、构件处理说明、屏幕设计文档、报表设计文档、程序设计文档、文件设计文档、数据库设计文档。

软件包括程序和文档两部分。信息系统的文档是系统建设过程的记录，是系统维护的重要依据，是开发人员与用户交流的工具。信息系统的文档，不但包括应用软件开发过程中产生的文档，还包括硬件采购和网络设计中形成的文档；不但包括上述有一定格式要求的规范文档，也包括系统建设过程中的各种来往文件、会议纪要、会计单据等资料形成的不规范文档；不但包括系统实施纪录，也包括程序资料和培训教程等。

系统规划和系统分析阶段的文档主要包括系统可行性研究报告、系统总体规划报告、

系统开发合同、系统方案说明书等。

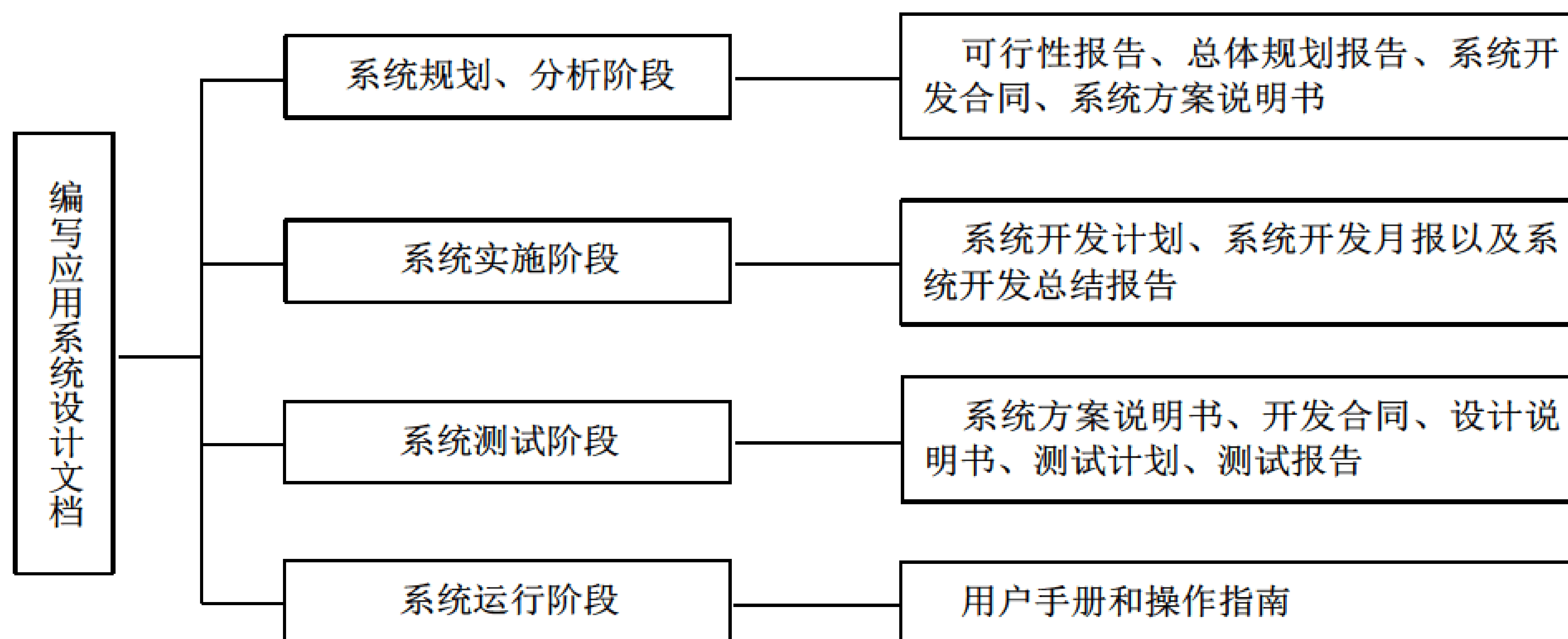


图 2-15　编写应用系统设计文档知识框图

系统实施阶段的文档主要有系统开发计划（包括工作任务分解表、网络图、甘特图、预算分配表等）、系统开发月报以及系统开发总结报告等项目管理文件。

系统测试阶段需要用到的文档主要有系统方案说明书、系统开发合同、系统设计说明书、测试计划、系统测试报告。系统测试人员再将评估结果撰写成系统测试报告。

系统运行阶段，用户通过系统开发人员撰写的文档运行系统，这里的文档主要是指用户手册和操作指南。

系统运行和维护阶段的文档主要有系统设计说明书和系统开发总结报告。有的开发总结报告写得很详细，分为研制报告、技术报告和技术手册 3 个文档，其中的技术手册记录了系统开发过程中的各种主要技术细节。

2．难点分析

一项计算机软件的筹划、研制及实现，构成一个软件开发项目。一个软件开发项目的进行，一般需要在人力和自动化资源等方面做重大的投资。为了保证项目开发的成功，最经济地花费这些投资，并且便于运行和维护，在开发工作的每一阶段，都需要编制一定的文件。这些文件连同计算机程序及数据一起构成计算机软件。文件是计算机软件中不可缺少的组成部分，它有以下作用。

① 作为开发人员在一定阶段内的工作成果和结束标志；

② 向管理人员提供软件开发过程中的进展和情况，把软件开发过程中的一些“不可见的”事物转换成“可见的”文字资料。以便管理人员在各个阶段检查开发计划的实施进展，使之能够判断原定目标是否已达到，还将继续耗用资源的种类和数量；

③ 记录开发过程中的技术信息，便于协调以后的软件开发、使用和修改；

④ 提供对软件的有关运行、维护和培训的信息，便于管理人员、开发人员、操作人员和用户之间相互了解彼此的工作；

⑤ 向潜在用户报导软件的功能和性能，使他们能判定该软件能否服务于自己的需要。

计算机软件产品开发文件编制指南（GB 8567—88）中指出：一项计算机软件的开发过程中，一般地说，应该产生 14 种文件。这 14 种文件是：可行性研究报告、项目开发计划、软件需求说明书、数据要求说明书、概要设计说明书、详细设计说明书、数据库设计说明书、用户手册、操作手册、模块开发卷宗、测试计划、测试分析报告、开发进度月报、项目开发总结报告。

2.7　设计评审

信息系统的评审分为广义和狭义两种。针对软件工程过程中的不同环节，设计评审的内容也不同，主要包括项目计划评审、软件需求评审、总体设计评审、详细设计评审、代码审查、系统测试计划评审、测试用例进行评审等。

如图 2-16 所示是本节的知识框图。

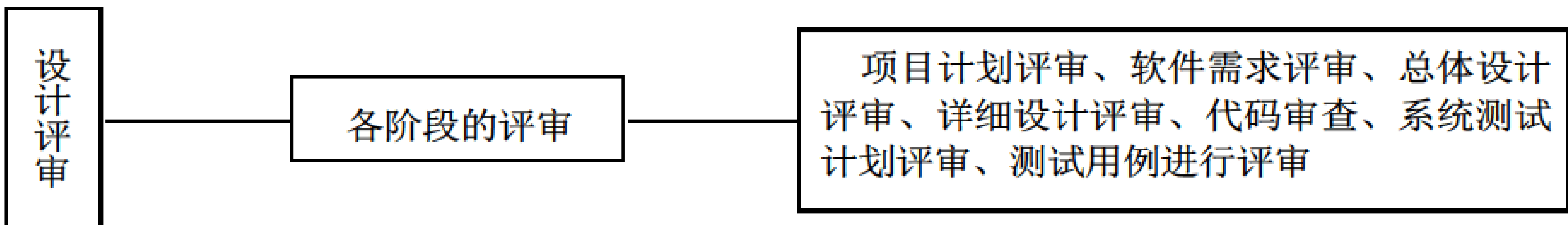


图 2-16　设计评审知识框图

1．知识点提炼

信息系统的评审分为广义和狭义两种。广义的信息系统评审是指从系统开发开始到结束的每一阶段都需要进行评审。狭义的信息系统评审是指在系统建成并投入运行之后所进行的全面、综合的评审。

系统评价指标包括系统质量、技术水平、运行质量、用户需求、系统成本、系统效益和财务评价。

广义的信息系统评审包括项目计划评审、软件需求评审、系统测试计划评审、总体设计评审、详细设计评审、代码审查和对测试用例进行评审。

软件需求评审主要是检查软件需求设计的完整性、正确性、一致性、可测试性等是否符合要求，需求文档是否齐全，是否符合有关标准规定。

总体设计评审主要审核目标系统的整体结构是否合理；各个功能模块间是否满足低耦合和高内聚；功能模块的作用范围是否在其控制范围内；所有已定义的软件需求是否均被所设计的系统覆盖；是否明确指出了系统各模块的功能、模块间的层次关系及接口控制特征；设计文档是否齐全，是否符合有关标准规定。

详细设计评审主要审核详细设计说明书与概要设计说明书是否一致；模块设计的质量（模块独立性，接口关系简单，规模适中，逻辑简单性，数据结构简单性）；设计文档是否

齐全，是否符合有关标准规定。

代码完成后可以有选择性的进行代码审查。

测试前期文档完成后对测试用例进行评审。

2．典型例题

【例题 2-32】 根据自己的项目经验，列举在项目各个阶段进行评审的内容，以及一般信息项目中评审时的注意点。

【解析】 一般的信息项目按阶段分为如下几方面的评审。

① 项目计划评审。

② 软件需求评审。在需求分析阶段结束后，要对该阶段工作结果即软件需求设计进行评审，主要是检查软件需求设计（软件需求说明书）的完整性、正确性、一致性、可测试性等是否符合合同或用户要求，还有就是看有关需求的文档是否齐全、是否符合有关标准规定。评审会上应做出结论，提出意见、建议，结论通常为合格、修改或重新设计。

③ 概要设计或者说总体设计评审。审核目标系统的整体结构是否具有好的形态，各功能模块间应满足低耦合度和高内聚度，功能模块的作用范围应在其控制范围内；所有已定义的软件需求是否均被所设计的系统覆盖；是否明确指出了系统各模块的功能、模块间的层次关系及接口控制特征；文件是否齐全，并符合有关标准规定。评审会上应做出结论，提出意见、建议，结论通常为合格、修改或重新设计。

① 详细设计评审。审核详细设计说明书与概要设计说明书是否一致；模块设计的质量（模块独立性，接口关系简单，规模适中，逻辑简单性，数据结构简单性）；文件是否齐全并符合有关标准规定。

② 代码完成后可以有选择性地进行代码审查（对关键模块，对水平比较低的程序员）。

③ 系统测试计划评审（初级的测试计划）。

④ 测试前期文档完成后对测试用例进行评审。

练习题

1．阅读以下有关数据库连接的叙述，回答问题 1 和问题 2。

在 Windows 环境下进行数据库访问工作有两种选择：使用 DAO 技术或者使用 ODBC 技术。ODBC（Open DataBase Connectivity）即开放式数据库互连，作为 Windows 开放性标准结构的一个重要部分已经为很多的 Windows 程序员所熟悉。DAO（Data Access Objects）即数据访问对象集是 Microsoft 提供的基于一个数据库对象集合的访问技术。它们都是 Windows API 的一个部分，可以独立于 DBMS 进行数据库访问。

【问题 1】 ODBC 和 DAO 的区别有哪些？

【问题 2】 在选择 ODBC 和 DAO 连接数据库时，有哪些好的建议？

2．一般来说，脚本语言只适用于数据表示。对于数据库访问，脚本语言是无能为力

了。在某数据库项目中，数据库平台采用了 SQL Server 2000 作为后端平台，采用的 B/S 模式，客户端的浏览器为 IE 5.0。如果希望在客户端通过 Javascript 获得数据库数据，请给出具体方案。

3．阅读以下有关 COM 和 DCOM 的叙述，回答问题 1 和问题 2。

在微软家族中，DCOM 是 COM 的一种。由于习惯的缘故，一般认为 COM 是在本地执行的 COM 组件，而 DCOM 是分布的 COM 组件，它在网络上的另一台计算机上执行。

【问题 1】 以上提到的本地 COM 和 DCOM 在存在形式、过程调用、效率、安全性和配置难易程度方面有哪些不同点？

【问题 2】 以上提到的本地 COM 和 DCOM 在调用形式上面有哪些相似之处？

4．阅读以下有关 ASP 应用开发的叙述，回答问题。

ASP 全称 Active Server Page，它提供了一个在服务器端执行脚本指令的环境（包括 HTML、VBScript 和 JavaScript 等），通过这种环境，用户可以创建和运行动态的 Web 应用程序。由于所有的程序都在服务器端执行，这样就大大减轻了客户端浏览器的负担，提高了交互速度。利用 ASP 不仅能够产生动态的、交互的、高性能的 Web 应用程序，而且可以进行复杂的数据库操作。

信息处组织了信息工程部有关管理人员和业务骨干，召开了 3 次小结与分析会议，在会上集中讨论了 ASP 有关的问题，在归纳的意见中出现了以下的一些内容。

① ASP 并不是一种语言，它所使用的语言通常是 VBScript 或者 JavaScript，通过这两种脚本语言，能够很方便地开发 ASP 应用程序。ASP 本身包含了 VBScript 和 JavaScript 的引擎，使得脚本可以直接嵌入 HTML 中，而且还可以通过 Active X 控件实现更为强大的功能。

② ASP 无须编译或链接即可直接解释执行，ASP 脚本集成于 HTML 中。但是由于 ASP 本身语法的特殊性，使用常规文本编辑器不能进行页面的设计，只有通过专门的开发工具如 Visual interdev 来编写。

③ ASP 独立于浏览器。用户端只要使用可解释常规 HTML 码的浏览器，即可浏览 ASP 所设计的主页。ASP 脚本是在站点服务器端执行的，因此，若不通过从服务器下载来观察 ASP 主页，在浏览器端将看不到正确的页面内容。但如果是在服务器上安装的浏览器，则不需要从服务器下载。

④ 在 ASP 脚本中可以方便地引用系统组件和 ASP 的内置组件，还能通过定制 ActiveX 服务器组件来扩充功能。与任何 ActiveX Scripting 语言兼容。

⑤ ASP 的源程序码不会外漏。ASP 脚本在服务器上执行，传到用户浏览器的只是 ASP 执行结果所生成的常规 HTML 码，这样可保证程序代码不会被他人盗取。

⑥ ASP 所完成的功能有：处理由浏览器传送到站点服务器的表单输入；访问和编辑服务器端的数据库表；使用浏览器即可输入、更新和删除站点数据库中的数据；读写站点服务器的文件，实现访客计数器。

⑦ ASP 所完成的功能有：取得浏览器信息管理等内置功能；读写用户端硬盘的各种文件，以记录用户的数据；可以实现在多个主页间共享信息，以开发复杂的商务站点应用程序。

⑧ 使用 VBScript 或 JavaScript 等简易的脚本语言，结合 HTML，快速完成站点的应用程序。通过站点解释器执行脚本语言，产生或更改在客户端执行的脚本语言。

⑨ 扩充功能的能力强，可通过使用多种程序语言制作的 ActiveX Server Component 以满足自己的特殊需要。

⑩ ASP 是通过一组统称为 ADO（ActiveX Data Objects）的对象模块来存取数据库，无论采用的是什么数据库，只要该数据库具有对应的 ODBC 或 OLE DB 驱动程序，ADO 对象就能加以存取。事实上，ASP 提供的 ADO 对象模块包含了 6 个对象和 3 个集合，比较常用的是 Connection、Recordset、Command、Field 等对象。

【问题】 工程师指出在上述 10 条意见中，有 3 条的提法是不够恰当的或者是不够全面的，请指出其序号，并各在 50 字以内说明理由。

5．某单位一信息项目的数据库平台采用了 SQL Server 2000 作为后端平台，采用 Client/Server 模式。由于和其他 Web 服务器共享存储空间，这个 SQL Server 只有很少的空余空间。为了节省客户的资金投入，所以任何新建的数据库都必须尽可能精确地计算容量，以减少对空间的浪费甚至做到不浪费空间。作为确定新数据库容量所需计算的一部分，数据库管理员搜集了以下资料：数据库将包含 20 个存储过程，每个存储过程占用从 77KB～103KB 空间。数据库将会包含两个视图，每个视图大约 12KB。但是在数据库进入实际工作状态后通过这两个视图可以显示大约 100MB 的数据。根据自己的项目经验，以提供的这些信息作为依据，这 20 个存储过程和两个视图需要占用多少空间？

6．某单位一信息项目的数据库平台采用了 SQL Server 7 作为后端平台，采用 Client/Server 模式。数据库管理员将服务器配制成 NT 验证模式后，用户无法使用他们的 Windows NT 登录名访问服务器。经过信息处技术人员的讨论以后，小组将问题定位到以下几个方面，事实上，只有两个选项是最可能导致这种情况发生的原因。根据自己的项目经验，你认为是哪两个方面？

① 数据库管理员没有重新启动管理所有安全请求的 SQL Server Agent 服务。

② 数据库管理员没有重新启动 SQL Server 服务。

③ 数据库管理员没有为任何用户授予对服务器进行管理的访问权。

④ 数据库管理员没有使用企业管理器对 Windows NT 账户的登录名进行映射。

练习题答案

1．**【解析 1】** ODBC 和 DAO 访问数据库的机制是完全不同的。ODBC 的工作依赖于数据库制造商提供的驱动程序，使用 ODBC API 的时候，Windows 的 ODBC 管理程序把数

据库访问的请求传递给正确的驱动程序，驱动程序再使用 SQL 语句指示 DBMS 完成数据库访问工作。DAO 则绕开了中间环节，直接使用 Microsoft 提供的数据库引擎（Microsoft Jet Database Engine）提供的数据库访问对象集进行工作，速度比 ODBC 快。数据库引擎目前已经达到了 3.0 版本。它是 DAO、MS Access、MS Visual Basic 等 Windows 应用进行数据库访问的基础。引擎本身的数据库格式为 MDB，也支持对目前流行的绝大多数数据库格式的访问，当然 MDB 是数据库引擎中效率最高的数据库。

【解析 2】 如果您使用 Client/Server 模式的话，建议使用 ODBC 方案；如果希望采用 MDB 格式的数据库，或者注重数据库引擎的速度，那么 DAO 是更好的选择。

2.**【解析】** 该方案可以通过 xml 数据岛实现。

利用 xmlDom 从服务器读出存放数据库数据的 xml 文件，这样便可以操作 xml 文件。在客户端动态显示，并可以进行一些特殊处理，可以实现许多在服务器端必须通过脚本并且需要多次连接的操作，比如分页、在查询结果中查询、排序、统计等等。例如，将一个 xml 文件从服务器端读出，然后再获得 xsl，这样经过 xslt 转换形成一个菜单等。

3.**【解析 1】** 不同点主要存在于以下几个方面。

COM 有两种存在形式，动态连接库和可执行程序，但 DCOM 必须是可执行程序。因为 DCOM 不可能在客户程序的内存空间运行，所以不能是动态连接库。

COM（动态连接库形式）可以不用 RPC 通信，而 DCOM 必须使用 RPC 远程调用。

COM（动态连接库形式）与客户共同存在于同一内存空间，调用速度快，DCOM 的速度只有 COM 的万分之一。

COM（动态连接库形式）的安全性不高，客户程序可以造成服务 COM 发生错误，DCOM 安全性高，原因也是 COM 与客户程序共用内存空间造成的。

COM 程序配置简单，DCOM 配置较复杂，毕竟 DCOM 牵涉到网络和安全性。

【解析2】 客户程序不必知道 COM 的存在形式，有统一的接口调用方式，客户程序甚至不知道 COM 对象的位置，可能在同一台计算机上，也可以在网络的另一台计算机上。

4.**【解析】** 3 条提法不够恰当或不够全面的意见更正如下。

第 2 条，ASP 无须编译或链接即可直接解释执行 ASP 脚本集成于 HTML 中。ASP 易于生成，由于 ASP 文件本身就是文本文件，所以使用常规文本编辑器即可进行页面的设计。

第 3 条，ASP 独立于浏览器。用户端只要使用可解释常规 HTML 码的浏览器，即可浏览 ASP 所设计的主页。ASP 脚本是在站点服务器端执行的，因此，若不通过从服务器下载来观察 ASP 主页，在浏览器端将看不到正确的页面内容。同样在服务器上安装的浏览器，同样需要服务器的解释执行，用户才能看到正确的页面内容，否则只能看到 ASP 的源代码。

第 7 条，ASP 所完成的功能有：取得浏览器信息管理等内置功能；由于安全方面的考虑，ASP 只能通过 Cookies 读写用户端的硬盘文件，以记录用户的数据，而不能随意读取客户端的任意文件；可以实现在多个主页间共享信息，以开发复杂的商务站点应用程序。

5.**【解析】** 在 SQL Server 2000 中，首先要注意的是：新的数据库对象通常都以盘区

为单位，而不是以页面为单位来创建。例如，假设一个新的存储过程实际上只占用 1KB，SQL Server 也会为这个存储过程保留一个盘区，也就是 64KB，即便这个盘区并不会被全部使用；SQL Server 不能在一个盘区中放置多于一个的对象。① 因为每个存储过程需要的空间多于 64KB 但是少于 128KB，所以每个存储过程将占用两个盘区（128KB 的存储容量），也就是说总共需要 2560 KB 数据库空间。② 因为所有视图需要的空间都少于 64KB，所以每个视图只需要一个盘区，也就是说两个视图共需要 128KB。所以，所有在问题中提到的对象共需要 2688KB 数据库空间。

6.**【解析】** 最有可能的是数据库管理员没有使用企业管理器对 Windows NT 账户的登录名进行映射或者数据库管理员没有重新启动 SQL Server 服务。

第 3 章　数据库应用系统实施

本章提示

本章根据大纲的要求，全面介绍了整个系统的配置与管理、常用数据库管理系统的应用、数据库应用系统的安装、数据库应用系统的测试以及培训与用户支持等主要知识点。在对典型例题详细分析之后，还给出了适量的练习题，以帮助读者加深对这些内容的理解和掌握。如图 3-1 所示是本章的知识框图。

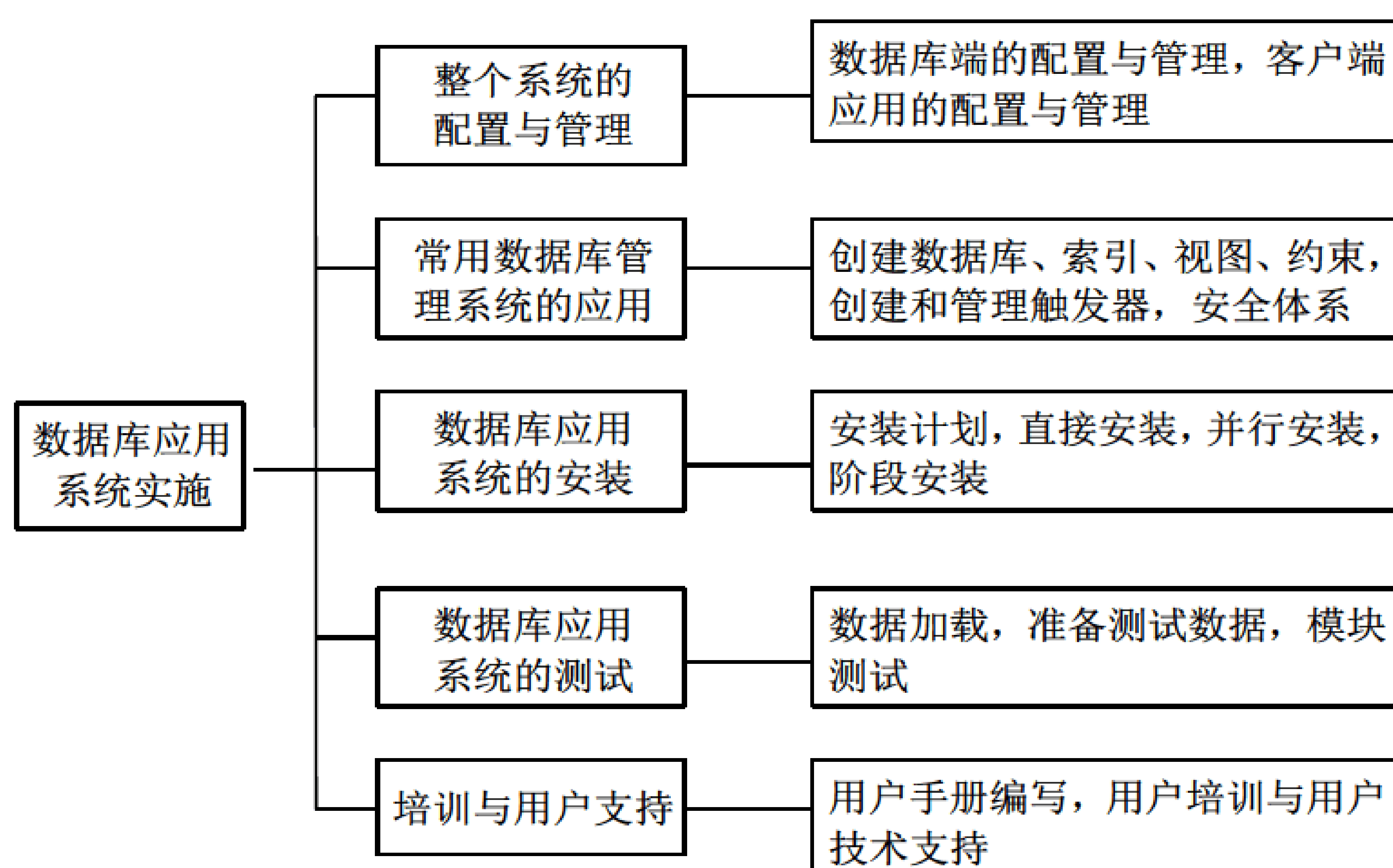


图 3-1　数据库应用系统实施知识框图

3.1　整个系统的配置与管理

整个系统包括数据库和数据库应用系统。配置和管理整个数据库应用系统是一个复杂的过程，本节分别讲述了数据库端的配置与管理和应用程序端的配置与管理。如图 3-2 所示是本节的知识框图。

1．知识点提炼

配置和管理整个数据库应用系统是一个复杂的过程，其中包含两个方面的内容：数据库端的配置和管理，以及应用程序端的配置和管理。

数据库端的配置和管理涉及如下内容。

运行数据库服务器（如果数据库是桌面数据库，则是本地的计算机）的硬件设备，包

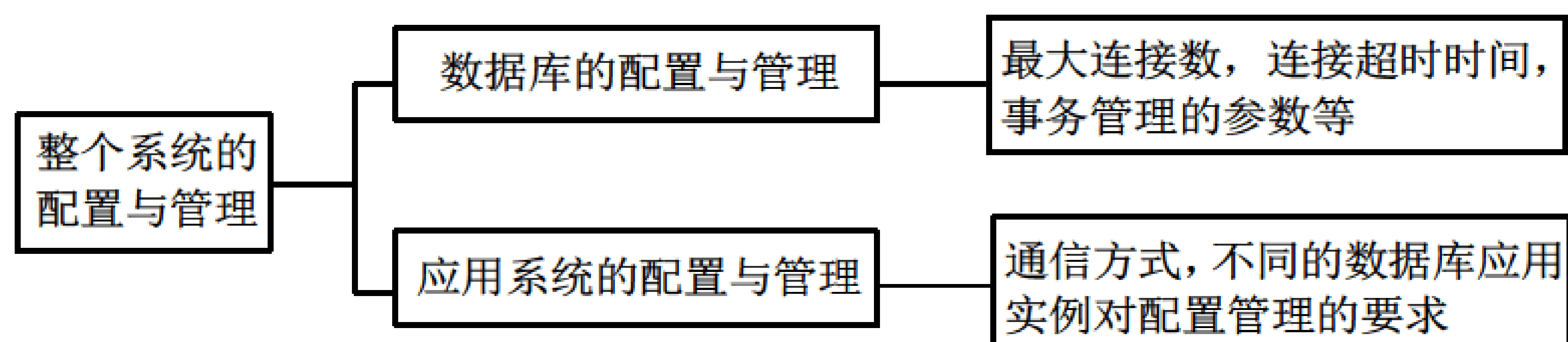


图 3-2　整个系统的配置与管理知识框图

括 CPU 的个数和频率，CPU 缓存的大小，内存的大小，内置硬盘的大小、个数和转速等；如果需要光纤存储，那么光纤存储接口卡的带宽大小也是其中的配置项之一；当然需要配置的硬件设备还包括 RAID 卡、网卡、后备电源、机架等。

数据库管理系统本身的配置，如数据库的最大连接数、数据库连接超时的时间、事务管理的参数等。

客户端应用程序和服务器之间通信的方式，例如 TCP/IP 方式、命名管道或是进程间共享内存的方式。在 Windows 环境下，应用程序与数据库（例如 SQL Server 2000 实例）在同一台计算机上，则可以使用 Windows 进程间通信（IPC）组件，例如以本地命名管道或共享内存的方式进行通信。如果应用程序在另外一台客户机上，则使用网络 IPC 与数据库进行通信。

客户端又分为胖客户端和瘦客户端。胖客户端包含有商业逻辑，其配置和管理稍微复杂，包括如下的几个方面：连接数据库服务器的驱动或连接程序，数据库服务器的名称、IP 地址和服务端口号，连接数据库的语言编码等。瘦客户端通常是一个浏览器，配置时主要对浏览器的连接协议（如 HTTP 或 HTTPs 等）、cookie 的存储方式、插件的支持方式（如 Java-applet 和 ActiveX 控件等）进行配置。

2．难点分析

配置数据库应用系统的过程本身并不复杂，但是要知道将某一个参数设置为多大的值才能满足应用的需求，这才是配置管理的真正难点。数据库应用用户对数据存储的要求非常复杂是造成数据库应用配置复杂的原因，下面是一些数据库应用实例对配置管理的要求。

在零售业的联机事务处理（OLTP）系统中，必须能够同时处理上千份订单。数据库的并发参数、数据库的连接参数、数据库的锁粒度等是配置这类系统的重点。

在大型 Web 站点中，数据库服务器和 Web 服务器常常紧密地结合在一起。顾客可以通过网络输入订单、联系服务部门和获取产品信息。这些网站需要着重对数据访问安全、与 Web 紧密集成的参数进行配置。

某类型的用户需要在没有联网的情况下继续工作。例如，出差在外的销售代表在某段时间不能与总公司的数据库进行网络的连接，可先使用笔记本中的数据与公司系统的当前数据保持同步，回到公司后将自己的现场工作结果合并到公司数据存储中。这一类应用可能对于数据库的同步复制参数配置提出了更高的要求。

管理人员和市场营销人员需要对公司数据中记录的趋势进行更为复杂的分析。他们需

要可靠的联机分析处理（OLAP）系统，这些系统能够很容易地通过 OLTP 数据生成，并支持复杂的数据分析。这就需要对多维数据库服务器进行配置。

对于独立的软件供应商（ISV）来说，可能要求数据存储系统由应用程序配置，然后系统自身可以自动调整，从而用户不需要专门的数据库管理员不间断地监视和调整应用程序。

3．典型例题

【例题 3-1】 在 Access 2000 数据库管理系统中，为什么有时打开数据库时，会得到“数据库被用户管理员锁定”的消息？如何处理这个问题？

【解析】 这个消息是说明网络上正有其他的用户以独占方式打开数据库。当 Access 没有设置数据库安全性时，任何打开数据库的人都称为“管理员”，所以会得到“数据库被用户管理员锁定”的消息。要处理这个问题，请按照以下步骤进行操作。

① 首先应找到独占数据库的用户。

② 让该用户选择“工具”/“选项”，单击“高级”标签，将“默认打开模式”从“独占”改为“共享”。

③ 退出 Access 再重新启动。这时，数据库不再锁定，其他用户可以同时使用该数据库。

【例题 3-2】 阅读以下有关数据库管理的叙述，回答问题 1 和问题 2。

在数据库应用的设计中，往往会需要获取某些表的记录总数，用于判断表的记录总数是否过大，是否需要备份数据等。通常的做法是：select count(*)as c from table_name 。然而对于记录数巨大的表，上述做法将会非常耗时。

【问题 1】 技术员小王提出如下方案：在表的某个字段上做聚簇索引；实验证明第一次执行该语句的时间和没有索引的时间差不多，之后执行上述语句速度变快。请分析这种方案的缺点有哪些。

【解析】 缺点主要有如下几点：

① 当表的记录数发生较大变化后，再执行该语句又会经历一次耗时的过程；

② 不是每个表都适合做聚簇索引的，对于数量巨大的表，如果需要经常增删操作，建聚簇索引是一个很不明智的做法，将会极大地影响增删速度。

【问题 2】 在 MS SQL Server 数据库中每个表都在 sysindexes 系统表中拥有至少一条记录，该记录中的 rows 字段会定时记录表的记录总数。如表 3-1 所示是 sysindexes 表相关记录的含义。

表 3-1　sysindexes 表的相关记录

列　名	数 据 类 型	描　　述
id	int	表 ID（如果 indid = 0 或 255）。否则为索引所属表的 ID
indid	smallint	索引 ID：0=表 1=聚簇索引>1=非聚簇索引 255=具有 text 或 image 数据的表条目。当表没有聚簇索引时，indid=0，否则为 1
rows	int	基于 indid=0 和 indid=1 的数据级行数，该值对于 indid>1 重复。如果 indid=255，rows 设置为 0

那么获取表的记录总数只需执行如下语句：select rows from sysindexes where id = object_id（tablename）and indid in（0,1）。该方法获取表的记录总数的速度非常快，在毫秒级就可以完成，相比 select count(*)要快上数倍。请分析这种方案的缺点有哪些。

【解析】 缺点主要有如下几点：

① 该方法得到的表的总记录数不是一个精确值，原因是 MS SQL Server 并不是实时更新该字段的值，而是定时更新；

② 该方案利用了数据库管理系统的一些功能，移植性不好。

【例题 3-3】 阅读以下有关信息系统需求分析的叙述，回答问题 1 和问题 2。

Oracle 数据字典是由表和视图组成的，存储有关数据库结构信息的一些数据库对象。数据字典描述了实际数据是如何组织的。对它们可以像处理其他数据库表或视图一样进行查询，但不能进行任何修改。

Oracle 数据字典通常是在创建和安装数据库时被创建的，Oracle 数据字典是 Oracle 数据库系统工作的基础，没有数据字典的支持，Oracle 数据库系统就不能进行任何工作。

【问题 1】 在 Oracle 数据字典中，许多视图都有 3 个不同的实例，它们的前缀分别为 USER_、ALL_及 DBA_。这些实例有什么区别和联系？

【解析】 USER_为前缀的数据字典视图通常记录执行查询的账户所拥有的对象的信息；ALL_为前缀的数据字典视图通常记录包括执行查询的账户所拥有的对象的信息及授权至 PUBLIC 的账户所拥有的对象的信息；DBA_为前缀的数据字典视图则包含所有数据库对象的信息，而不管其所有者。

【问题 2】 Oracle 中还包括其他的字典视图，主要有 V$视图，之所以这样叫是因为它们都是以 V$或 GV$开头的。概括说明 V$视图。

【解析】 V$视图是基于 X$虚拟视图的。V$视图是 SYS 用户所拥有的，在默认的情况下，只有 SYS 用户和拥有 DBA 系统权限的用户可以看到所有的视图，没有 DBA 权限的用户可以看到 USER_和 ALL_视图，但不能看到 DBA_视图。与 DBA_、ALL_和 USER_视图中面向数据库信息相反，这些视图可视地给出了面向实例的信息。

【例题 3-4】 阅读以下有关 RAID 的叙述，回答问题 1 到问题 3。

RAID，为 Redundant Arrays of Independent Disks 的简称，中文为独立冗余磁盘阵列。作为高性能的存储系统，已经得到了越来越广泛的应用。RAID 的级别从 RAID 概念的提出到现在，已经发展为 6 个级别，分别是 0、1、2、3、4、5。虽然 RAID 不是 Microsoft SQL Server 的一部分，但它的实现直接影响 SQL Server 的性能。SQL Server 一般使用 RAID 等级 0、1 和 5。

【问题 1】 根据你自己的项目经验，谈谈 RAID 0 的构成和优缺点。

【解析】 将多个较小的磁盘合并成一个大的磁盘，不具有冗余，并行 I/O，速度最快。RAID 0 亦称为带区集。它是将多个磁盘并列起来，成为一个大硬盘，在存放数据时，将数据按磁盘的个数来进行分段，然后同时将这些数据写进这些磁盘中。

所以，在所有的级别中，RAID 0 的速度是最快的。但是 RAID 0 没有冗余功能，如果一个磁盘（物理）损坏，则所有的数据都无法使用。

【问题 2】 根据你自己的项目经验，谈谈 RAID 1 的构成和优缺点。

【解析】 两组相同的磁盘系统互作镜像，速度没有提高，但是允许单个磁盘出错，可靠性最高。RAID 1 就是镜像。其原理为在主硬盘上存放数据的同时也在镜像硬盘上写相同的数据。当主硬盘（物理）损坏时，镜像硬盘则代替主硬盘的工作。因为有镜像硬盘的数据备份，所以 RAID 1 的数据安全性在所有的 RAID 级别上来说是最好的。但是其磁盘的利用率却只有 50%，是所有 RAID 上磁盘利用率最低的一个级别。

【问题 3】 根据你自己的项目经验，谈谈 RAID 5 的构成和优缺点。

【解析】 向阵列中的磁盘写数据，奇偶校验数据存放在阵列中的各个磁盘上，允许单个磁盘出错。RAID 5 也是用数据的校验位来保证数据的安全，但它不是用单独硬盘来存放数据的校验位，而是将数据段的校验位交互存放于各个硬盘上。这样，任何一个硬盘损坏，都可以根据其他硬盘上的校验位来重建损坏的数据。

【例题 3-5】 阅读以下有关航空订票系统的叙述，回答问题。

某单位一信息项目——航空订票系统的数据库平台采用了 SQL Server 2000 作为后端平台，前端选择了 Microsoft 公司的 Visual Basic 及 Visual Modeler 作为开发工具，采用 Client/Server 模式。

该信息系统属于联机事务处理系统（OLTP）。该系统有大量的用户同时执行更改实时数据的事务。尽管用户对数据的单个请求一般只引用少量记录，但是，这些请求有许多是同时发生的。在这种类型的应用程序中，主要关心的是并发性和原子性。

【问题】 为了提高数据库应用的性能，该联机事务处理设计的注意事项有哪些？

【解析】 该联机事务处理设计的注意事项有如下几方面。

数据物理存放方式。对于 OLTP 系统，输入/输出瓶颈是一个尤其应该关心的问题，原因在于修改整个数据库中数据的用户很多。确定数据的可能访问模式，并将经常访问的数据放在一起。在此过程中，可辅以文件组和 RAID（独立磁盘冗余阵列）系统。

缩短事务处理时间，将长期锁减至最少，提高并发性。在事务处理期间，避免用户交互。无论何时，只要有可能，就通过执行单个存储过程来处理整个事务。在事务内对表的引用顺序可能会影响并发性。将对经常访问的表的引用置于事务的末尾，以便将控制锁的持续时间减至最短。

联机备份是应该考虑的。OLTP 系统通常的特征是连续操作（一天 24 小时，一周 7 天），为达到此目的，停工时间要保持绝对最短。尽管 SQL Server 2000 可以在数据库正在使用时对其进行备份，但是应将备份过程安排在活动不频繁时进行，以使对用户的影响减至最小。

数据库的高度规范化。尽可能减少冗余信息以提高更新的速度，从而提高并发性。减少数据还可以提高备份的速度，因为只需要备份更少的数据。

处理很少或没有历史的聚合数据时，可以将很少引用的数据归档到单独的数据库中，或者从经常更新的表中移出，并置于仅含历史数据的表中。这将保持表尽可能地小，从而缩短备份时间，改善查询性能。

小心使用索引。每次添加或修改行时，必须更新索引。若要避免对经常更新的表进行过多的索引，索引范围应保持较窄。请用索引优化向导设计索引。

OLTP 系统需要最佳的硬件配置以处理较大数目的并发用户并加快响应时间。

3.2 常用数据库管理系统的应用

目前常用的数据库管理系统有 SQL Server、Oracle、Sybase、DB2、Visual FoxPro 和 Access 等。本节主要讲述了如何创建数据库、创建表、创建索引、创建视图、创建约束、创建 UDDT（用户自定义类型）、创建和管理触发器等，还讲述了如何创建安全体系。如图 3-3 所示是本节的知识框图。

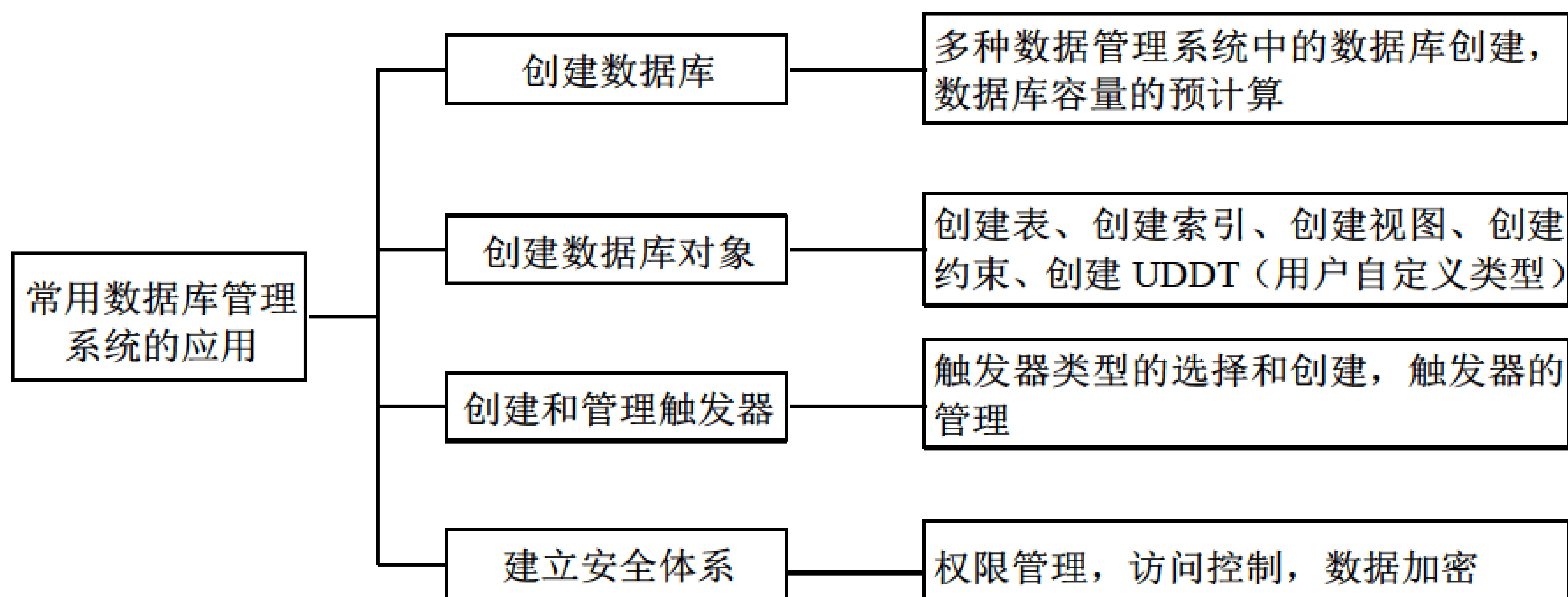


图 3-3 常用数据库管理系统的应用知识框图

1．知识点提炼

目前，常用的数据库管理系统有 SQL Server、Oracle、Sybase、DB2、Visual FoxPro 和 Access 等。下面以 SQL Server 为主线进行讲解。

（1）创建 SQL Server 数据库

在 SQL Server 中，数据库是由数据库文件和事务日志文件组成的。一个数据库至少应包含一个数据库文件和一个事物日志文件。

数据库文件（Database File）是存放数据库数据和数据库对象的文件。一个数据库可以有一个或多个数据库文件，一个数据库文件只属于一个数据库。当数据库中有多个数据库文件时，其中有一个文件被定义为主数据库文件（Primary Database File），扩展名为 mdf，主数据库文件用来存储数据库的启动信息和部分或全部数据。一个数据库只能有一个主数据库文件。其他数据库文件被称为次数据库文件（Secondary Database File），扩展名为 ndf，

用来存储主文件没存储的其他数据。

采用多个数据库文件来存储数据有以下的优点。

① 数据库文件可以不断扩充而不受操作系统文件大小的限制。

② 可以将数据库文件存储在不同的硬盘中，这样可以同时对几个硬盘进行数据存取，提高了数据处理的效率，这对于服务器型的计算机尤为有用。

事务日志文件（Transaction Log File）是用来记录数据库更新情况的文件，扩展名为 ldf，比如，使用 INSERT、UPDATE、DELETE 等对数据库进行更改的操作都会记录在日志文件中，而查询语句 SELECT 等对数据库内容没有影响的操作则不会记录。一个数据库可以有一个或多个事务日志文件。

在 SQL Server 中采用“Write-Ahead（提前写）”方式处理事务，即对数据库的修改先写入事务日志中，再写入数据库，其具体操作如下。

① 系统将更改操作写入事务日志中。

② 更改存储在计算机缓存中的数据。为了提高执行效率，此更改不会立即写到硬盘中的数据库里，而是由系统以固定的时间间隔执行 CHECKPOINT 命令，将更改过的数据批量写入硬盘。

在 SQL Server 中，当执行数据更改时会设置一个开始点和一个结束点，如果尚未到达结束点就由于某种原因而使操作中断，则在 SQL Server 重新启动时系统会自动还原已修改的数据使其返回到未被修改的状态。由此可见，当数据库系统遭到破坏时，可以用事务日志还原数据库内容。

文件组（File Group）是将多个数据库文件集合起来形成的一个整体，每个文件组有一个组名。与数据库文件一样，文件组也分为主文件组和次文件组。一个文件只能存在于一个文件组中，一个文件组也只能被一个数据库使用。

主文件组中包含了所有的系统表。当建立数据库时，主文件组包括主数据库文件和未指定组的其他文件。在次文件组中可以指定一个默认文件组，那么当创建数据库对象时，如果没有指定将其放在哪一个文件组中，就会将它放在默认文件组中。如果没有指定默认文件组，则默认主文件组为默认文件组。

在 SQL Server 中，可以用 Query Analyzer 创建数据库：

CREATE DATABASE 数据库名

还可以用 Enterprise Manager 创建数据库，由于操作简单，在此不再赘述。

需要注意的是，在 SQL Server 中，数据库的名称最长为 128 个字符，且不区分大小写。一个服务器在理论上可以管理 32 767 个数据库。

（2）创建 Oracle 数据库

在 Oracle 数据库中，数据库系统由处理数据文件的一组程序组成。Oracle 数据库中存放两种类型的信息：一是用户数据，即特定应用程序的数据；二是系统数据，即数据库系统管理自身所需的数据，比如与特定数据库有关的数据文件的名称及存放地点。Oracle 数

据库由两种类型的文件组成：一是构成表空间的数据文件；二是事务日志文件。

表空间（Tablespace）是一个或多个数据文件的集合，所有的数据库对象都存储在表空间中。之所以称其为表空间，是因为它存放的数据库对象主要是表。

在大多数数据库中，下列表空间是必备的或者是常见的。

① 系统表空间。系统表空间中保存用于管理 Oracle 系统自身及其中存放的数据所需的信息，这些表空间的名称是固定的。

② 临时表空间。临时表空间是 Oracle 中临时使用的区域。当特定事件发生时，Oracle 用临时表空间管理有关的事务。

③ 工具表空间。工具表空间用来保存那些在 Oracle 数据库上运行的工具软件所需的对象。

④ 用户表空间。用户表空间用来存放用户专用的数据库对象。

⑤ 回滚表空间。回滚表空间用来存放数据库对象的回滚段。

⑥ 数据和索引表空间。索引是数据库对象的一个特殊类型。Oracle 使用索引进行快速数据检索。数据和索引表空间用来存放用户的应用数据。

除了与表空间联系紧密的数据文件外，Oracle 还有另一个与其相关的称做联机重做日志（Online Redo Log）的操作系统文件，也称做事务日志（Transaction Log）。Oracle 在这些特殊的操作系统文件中记录针对数据库进行的修改操作或事务。对数据库所做的所有修改工作都在内存中进行，之所以这样处理主要是出于性能方面的考虑，因为在磁盘 I/O 中操作比在内存中进行相应操作要慢得多。事务日志中总是保留所有事务的一个拷贝，这样 Oracle 可节省将内存中修改数据操作写回源数据文件所需的时间，保存有修改情况的最终拷贝将写回到物理的数据文件中。因为所有的处理都记录在事务日志中，因此，数据库系统可以使用这些事务记录进行恢复操作。对每一个 Oracle 数据库都要求至少具有两个事务日志。

Oracle 数据库可以在下列两种模式下运行：一是 ARCHIVELOG 模式。在该模式下将保存所有的事务日志。二是 NOARCHIVELOG 模式。在该模式下不保存旧事务日志。

控制文件是一个非常小的文件，其中存放一些与 Oracle 数据库所有文件相关的关键信息。Oracle 系统通过控制文件保持数据库的完整性以及决定恢复数据时使用哪些事务日志。数据库结构的所有修改都被记录在控制文件中。每个数据库至少有两个控制文件，建议用户最少生成两个控制文件，并分别保存在不同的磁盘上。

Oracle 实例（Instance）是有自己的系统全局区及其相关数据库文件的 Oracle 服务器进程集，它是访问 Oracle 数据库所需的一部分计算机内存和辅助处理进程。

（3）创建 Visual FoxPro 数据库

在 Visual FoxPro 中，关系数据库是以二维表的形式组织起来的数据集合。数据库（关系）可以分为库结构（关系模式）和数据记录（元组）两部分。在 Visual FoxPro 中，变量分字段变量和内存变量两大类。

字段变量依赖于数据库文件，共有 7 种类型：C 型、N 型、F 型、D 型、L 型、M 型和 G 型。其中 D 型、L 型、M 型的固定宽度分别为 8、1、10，M 型、G 型存储在与数据库主名同名的.FPT（备注）文件中。

内存变量独立于数据库文件，存储在内存中，共有 6 种类型：C 型、N 型、F 型、D 型、L 型和 S 型。可用 M→内存变量与同名的字段变量相区分。

在 Visual FoxPro 中，创建数据库的步骤如下。

① 建立库结构。用 CREATE 命令或单击菜单“文件”→“新建”，在“文件类型”中选择“表”，然后单击“新建文件”按钮。单击“保存”按钮存盘，按 Esc 键或单击“取消”按钮则可以放弃存盘。

② 将库结构保存后直接输入库记录即可。

③ 输入备注型字段数据。在 memo 处双击鼠标或者按组合键 Ctrl + PgDn 或 Ctrl + PgUp 打开 memo 窗口。

打开数据库的方法有两种。

① 使用 USE 命令。其语法格式如下：

USE 数据库名

② 使用菜单命令。单击菜单“文件”→“打开”，在“文件类型”中选择“表（*.dbf）”，选择盘符、文件名，然后单击“确定”按钮。

关闭数据库的方法有两种。

① 使用 USE 命令。

② 使用菜单命令。单击菜单“文件”→“关闭”即可关闭数据库。

修改数据库的方法分别如下。

① 修改数据库结构。修改库结构有两种方法：一是用 MODIFY STRUCTURE 命令；二是单击菜单“数据库”→“设置”→“修改”命令。

② 添加库记录。添加库记录的方法有 4 种：一是用 APPEND[blank] 命令；二是单击菜单“记录”→“添加”命令手工追加；三是用 APPEND FROM 命令从其他数据库追加记录；四是单击菜单“数据库”→“添加”，从其他数据库追加记录。

③ 插入库记录。插入库记录可以用 INSERT[blank][before]命令。

④ 修改库记录。修改库记录的方法有两种：一是用 CHANGE/EDIT 命令；二是单击菜单“记录”→“修改”命令。需要注意的是默认的修改范围为 REST。

⑤ 浏览数据库。浏览数据库的方法有两种：一是用 BROWS 命令；二是单击菜单“记录”→“浏览”命令。

⑥ 替换记录。替换记录的方法有两种：一是用 REPLACE…WITH 命令；二是单击菜单“记录”→“替换”命令。需要注意默认的替换范围为 NEXT 1。

⑦ 删除和恢复记录。删除和恢复记录包括逻辑删除和物理删除。

逻辑删除记录的方法有两种：一是用 DELETE 命令；二是单击菜单“记录”→“删除”

命令给记录打*号。默认的删除记录的范围是 NEXT 1。用 SET DELETE OFF/ON 命令可以设置逻辑删除是否有效，即*号标记的记录是否参与操作。逻辑删除还可以恢复，恢复逻辑删除的方法有两种：一是用 RECALL 命令；二是单击菜单“记录”→“恢复”命令将打*的记录恢复。默认的逻辑删除恢复范围为 NEXT 1。

物理删除记录的方法有两种：一是用 PACK 命令；二是单击菜单“数据库”→Pack 命令，将打*的记录真正删除。

清除记录的方法有两种：一是用 ZAP 命令；二是用 DELETE ALL 和 PACK 两条命令。

（4）创建表、创建索引、创建视图、创建约束、创建 UDDT（用户自定义类型）

在 SQL Server 中，数据表是关系数据库的基本组成单位，它物理地存储于数据库的存储文件中。表（TABLE）是存放用户数据的数据库对象。每个表的信息都放在数据字典中，使用数据字典来保证将正确类型的数据放到表中。

表包括基本表和视图。

基本表（BASE TABLE）是独立存在的表，不是由其他的表导出的表。一个关系对应一个基本表，一个或多个基本表对应一个存储文件。

视图（VIEW）是一个虚拟的表，是从一个或几个基本表导出的表。它本身不独立存在于数据库中，数据库中只存放视图的定义而不存放视图对应的数据，这些数据仍存放在导出视图的基本表中。当基本表中的数据发生变化时，从视图中查询出来的数据也随之改变。

数据表主要包括以下几个组成部分。

① 字段名，即列名。字段名最长为 128 个字符。字段名可包含中文、英文字母、下划线、#、货币符号（￥）及@。因为列名与属性相对应，所以同一表中不允许有重名的列。

② 字段数据类型。

③ 字段的长度、精度和小数位数。

在 SQL 语言中，使用语句 CREATE TABLE 创建数据表，创建数据表的 SQL 语法格式为：

CREATE TABLE <表名>（<列定义>[{，<列定义>|<表约束>}]）

其中，<表名>是合法标识符，最多可有 128 个字符，表名不允许重复。

<列定义>的语法格式为：

<列名><数据类型>[DEFAULT] [{<列约束>}]

其中的 DEFAULT 字段，若是某字段设置有默认值，当该字段未被输入数据时，则将该默认值自动填入该字段。

在 Visual FoxPro 中，用户定义数据类型（User-Defined Data Type，UDDT）是由用户特殊指定数据存放的数据类型。目前，数据库对象（如表）能够容纳更多的数据类型。表中的数据既可以以简单的数据类型存放，如 char、number、long 等等，也可以由用户自定义数据类型。在关系型数据库管理系统（ORDBMS）中，用户可以定义附加类型的数据，来指定数据结构和其上的操作。用户自定义数据类型使得开发复杂的数据如图像、声音和视频等变得更为容易。

从用户观点来看，基本表和视图都是关系。但由于视图是虚表，它并不对应一个存储的数据文件，因此通过视图对数据的修改要受到一定的限制。建立视图有两个作用：其一是可以简化查询命令，其二是可以限制某些用户的访问范围。

在 Visual FoxPro 中，定义视图的语法格式如下：

CREATE VIEW＜视图名称＞（属性名 1，属性名 2，…）

AS SELECT 查询模块

［WITH CHECK OPTION］；

取消视图的语法格式如下：

DROP VIEW ＜视图名称＞；

需要注意，取消视图后，其定义和以它为基础建立的其他视图将自动删除。

建立索引的语法格式如下：

CREATE［UNIQUE］INDEX＜索引文件名＞

ON ＜表名＞（索引关键字 ASC | DESC）；

其中 UNIQUE 为可选项，表示每一个索引关键字的值只对应唯一的元组。

取消索引的语法格式如下：

DROP INDEX ＜索引文件名＞；

更新数据的语法格式如下：

UPDATE ＜表名＞

SET ＜更新表达式＞

［WHERE＜条件＞＝；

删除数据时，删除的单位是元组，而不是元组的部分属性。一次可以删除一个元组、几个元组，以至于将整个表删成空表，只保留表的结构定义。删除同名属性时要注意保持数据的一致性。

删除数据的语法格式如下：

DELETE ＜元组＞

FROM ＜表名＞

WHERE ＜条件＞；

在 Access 中，创建表的步骤如下。

① 在数据库窗口中，单击“表”按钮。

② 单击工具栏的“新建”按钮。

③ 在新建表对话框中，选择“设计视图”，并单击“确定”按钮，在设计视图中将打开一个表设计窗口。如果选择表向导，Access 将启动一个向导，引导使用者一步步地创建表。表向导提供了一个样表作为新创建的表的基础。

④ 在表设计窗口上半部分的每一行中，输入字段名称、数据类型以及一个有助于以后识别这个字段的长描述（字段名和数据类型是必需的，而描述可选）。应尽量保持字段名

简短，使它们容易在窗体和报表中处理。

⑤ 在定义每个字段时，字段的属性会显示在表窗口的下半部分（可以通过按 F6 键或单击窗口中的相应位置切换到窗口上半部或下半部）。单击想设置的属性，从列表框中选择相应的设置。

⑥ 如果需要，可以创建一个主关键字（主关键字是一个字段或字段的组合，表中每个记录的主键值是唯一的）。单击选定字段的行选定器（每行左边的小方块），或按住鼠标左键在行选定器上拖动，以选择多个字段。然后，单击工具栏上的主关键字按钮，或单击菜单“编辑”→“主关键字”。如果在表中没有创建主关键字，Access 会提示是否要创建一个。如果选择“是”，Access 将创建一个自动编号字段，把它作为主关键字，如果选择“否”，将不创建主关键字。

⑦ 当完成表中字段的定义后，选择菜单“文件”→“保存”。为表输入一个名字，并单击“确定”按钮。

表是数据的基础，如果定义了一个表，也就定义了用于保存数据的字段。每个字段包含单一类型的信息，如地址、姓名、电话号码等。表含有很多记录，每个记录都包含一个实体的完整信息。

（5）创建和管理触发器

数据库触发器（DATABASE TRIGGERS）是存储在数据库中的程序，并当某一事件（如数据变化、用户登录等）发生时运行。数据库触发器与数据库表相关联。当事件发生时，触发器是存储在数据库中的过程，将被运行或触发。当表上的一个操作执行时，触发器被激活。当确定表上的操作将激活一个程序时，可以创建数据库触发器。当一个数据库触发器创建后，要指定数据库触发器在事务前还是事务后触发。

使用数据库触发器的主要目的是将数据存储到数据库之前检查其合法性。动作类型包括插入、删除和修改。使用触发器可以实现以下功能。

① 在数据存储到数据库之前对其进行检查。

② 数据进入时在其他表中生成记录。

③ 用触发器实现商业规则。该功能只能在某些类型的数据库管理系统或者某些更高版本中使用，考生应该掌握。

实现触发器时要特别注意，如果触发器没有按预期的目标执行，可能会在应用或用户访问数据库会话中产生不可预料的结果。

2．难点分析

目前常用的关系型数据库管理系统大部分都可根据在创建数据库时所定义的增长参数，自动扩充数据库。通过在现有的数据库文件上分配其他的文件空间，或者在另一个新文件上分配空间，还可以手动扩充数据库。如果现有的文件已经充满，则可能需要扩充数据或事务日志空间。如果数据库已经用完分配给它的空间而又不能自动增长，则会出现严重的错误。

在创建数据库时，需要估计填入数据时的数据库大小。估计数据库的大小有助于确定下列各项所需要的硬件配置。

① 达到应用程序的性能要求。

② 确保有适当的物理磁盘空间以存储数据和索引。

估计数据库的大小还可以帮助确定数据库设计是否需要进行精简处理。如果数据库的估计值太大，无法在单位中实现，则需要进行更多的规范化处理。相反，估计值可能比所期望的更小，这样就可以降低数据库的规范化程度以提高查询性能。

估计数据库的大小时，可以分别估计每个表的大小，然后累加所得的值。值得注意的是，表的大小还取决于表是否有索引和类型的索引。

3．典型例题

【例题 3-6】 阅读以下关于 Oracle 视图方面的叙述，回答问题 1 和问题 2。

Oracle 数据库作为大型数据库管理系统，近年来一直占有世界上高端数据库的最大份额，在 Oracle 数据库中，视图是原始数据库数据的一种变换，是查看表中数据的另外一种方式。可以将视图看成是一个移动的窗口，通过它可以看到感兴趣的数据。视图是从一个或多个实际表中获得的，这些表的数据存放在数据库中。那些用于产生视图的表叫做该视图的基表。一个视图也可以从另一个视图中产生。

视图的定义存在数据库中，与此定义相关的数据并没有再存一份于数据库中。通过视图看到的数据存放在基表中。视图看上去非常像数据库的物理表，对它的操作同任何其他的表一样。当通过视图修改数据时，实际上是在改变基表中的数据；相反，基表数据的改变也会自动反映在由基表产生的视图中。由于逻辑上的原因，有些视图可以修改对应的基表，有些则不能（仅仅能查询）。

【问题 1】 在 Oracle 系统中建立视图有哪些主要的好处？

【解析】 在 Oracle 系统中建立视图有如下几方面的好处。

简单性。看到的就是需要的。视图不仅可以简化用户对数据的理解，也可以简化他们的操作。那些被经常使用的查询可以被定义为视图，从而使得用户不必为以后的操作每次指定全部的条件。

安全性。通过视图用户只能查询和修改他们所能见到的数据。数据库中的其他数据既看不见也取不到。数据库授权命令可以使每个用户对数据库的检索限制到特定的数据库对象上，但不能授权到数据库特定的行和特定的列上。通过视图，用户可以被限制在数据的不同子集上。

逻辑数据独立性。视图可帮助用户屏蔽真实表结构变化带来的影响。

【问题 2】 视图可以使应用程序和数据库表在一定程度上独立。如果没有视图，应用一定是建立在表上的。有了视图之后，程序可以建立在视图之上，从而程序与数据库表被视图分割开来。请举例说明视图可以在哪些方面使程序与数据独立。

【解析】 视图使程序与数据独立主要体现在如下的几个方面。

如果应用建立在数据库表上，当数据库表发生变化时，可以在表上建立视图，通过视图屏蔽表的变化，从而应用程序可以不动。

如果应用建立在数据库表上，当应用发生变化时，可以在表上建立视图，通过视图屏蔽应用的变化，从而使数据库表不动。

如果应用建立在视图上，当数据库表发生变化时，可以在表上修改视图，通过视图屏蔽表的变化，从而应用程序可以不动。

如果应用建立在视图上，当应用发生变化时，可以在表上修改视图，通过视图屏蔽应用的变化，从而数据库可以不动。

【例题 3-7】 阅读以下有关 Access 2000 安全体系的叙述，回答问题 1 和问题 2。

Access 2000 在桌面数据库的应用很广泛，但是 Access 2000 的安全性一直令人担忧。例如：Access 的安全性提供了口令保护，但是仍然可以用 Visual Basic 或其他前端应用程序取得数据。

【问题 1】 如何处理上文中提到的问题？

【解析】 Access 的安全性提供了口令保护，但是不能防止使用实用程序读取其中的数据或用其他软件打开数据库。为了提供这一级别的保护，需要对数据库进行加密（Encrypt）。加密将数据库文件以编码格式保存，只能由 Access 读取。对数据库进行加密的操作如下。

① 关闭所有打开的数据库。

② 选择“工具”→“安全”→“加密/解密数据库”。

③ 在打开的对话框中，选择要加密的数据库并单击“确定”按钮。

④ 为加密后的数据库指定一个文件名和位置（驱动器和文件夹），然后单击“保存”按钮。

【问题 2】 操作加密后的数据库和未加密的数据库的效率有什么区别？为什么？

【解析】 需要注意的是，Access 对加密的数据库进行操作要比普通数据库慢，因为 Access 必须把加密的数据解密以后才能读取。

【例题 3-8】 阅读以下有关存储过程开发的叙述，回答问题。

某单位一信息项目的数据库采用了 SQL Server 作为后端平台，前端选择了 Borland 公司的 Delphi 作为开发工具，采用 Client/Server 模式。在数据库的开发过程中，项目组遇到了如下问题。项目中有大量复杂的业务逻辑和对数据库的操作，这个时候就会用存储过程来封装数据库操作。但是项目实施最初，开发人员没有经验，项目的存储过程较多，书写又没有一定的规范，所以项目后期的系统维护困难加大，大存储过程逻辑又难以理解。幸好项目经理及时发现这个问题，召集所有技术人员开会，就存储过程的开发提出以下一些看法。

① 开发人员如果用到其他库的 TABLE 或 VIEW，务必在当前库中建立 VIEW 来实现跨库操作，最好不要直接使用 database.dbo.table_name，因为存储过程_depends 不能显示出

该存储过程所使用的跨库 TABLE 或 VIEW，不方便校验。

② 开发人员在提交存储过程前，应该使用 SET showplan ON 分析查询计划，做过自身的查询优化检查。

③ 尽量避免大事务操作，慎用 HOLDLOCK 子句，提高系统并发能力。

④ 尽量避免反复访问同一张或几张表，尤其是数据量较大的表，可以考虑先根据条件提取数据到临时表中，然后再做连接。

⑤ 尽量避免使用游标，因为游标的效率较差。如果游标操作的数据超过 1 万行，那么就应该改写；如果使用了游标，就要尽量避免在游标循环中再进行表连接的操作。

⑥ 注意 WHERE 子句写法，必须考虑语句顺序，应该根据索引顺序、范围大小来确定条件子句的前后顺序，尽可能地让字段顺序与索引顺序相一致，范围从大到小。

⑦ 尽量在 WHERE 子句中的=左边进行函数、算术运算或其他表达式运算，使系统充分使用索引。

⑧ 尽量使用 EXISTS 代替 SELECT COUNT（1）来判断是否存在记录，COUNT 函数只有在统计表中所有行数时使用，而且 COUNT（*）比 COUNT（1）效率更高。

⑨ 尽量使用>=，不要使用>。

⑩ 注意一些 OR 子句和 UNION 子句之间的替换。

注意表之间连接的数据类型，避免不同类型数据之间的连接。

注意存储过程中参数和数据类型的关系。

注意 INSERT、UPDATE 操作的数据量，防止与其他应用冲突。如果数据量超过 200 个数据页面（400 KB），那么系统将会进行锁升级，页级锁会升级为表级锁。

【问题】 在上述 13 条叙述中有 3 条是不正确或不恰当的，请指出其序号，并各在 50 字以内简要说明理由。

【解析】 如下 3 条是不正确或不恰当的。

第 7 条，不要在 WHERE 子句中的=左边进行函数、算术运算或其他表达式运算，否则系统将可能无法正确使用索引。

第 8 条，尽量使用 EXISTS 代替 SELECT COUNT（1）来判断是否存在记录，COUNT 函数只有在统计表中的行数时使用，而且 COUNT（1）比 COUNT（*）效率更高。

第 3 条，尽量避免大事务操作，慎用 HOLDLOCK 子句，提高系统并发能力。

【例题 3-9】 阅读以下有关触发器的叙述，回答问题 1 和问题 2。

在数据库应用开发中，有两种方法可设定自动化的资料处理规则，一种是条件约束，一种是触发器。一般而言，条件约束比触发器较容易设定及维护，且执行效率较好，但条件约束只能对资料进行简单的栏位检核，当涉及到多表操作等复杂操作时，就要用到触发器了。触发器有 AFTER 触发器和 INSTEAD OF 触发器。

AFTER 触发器：触发时机在资料已变动完成后，它将对变动资料进行必要的善后与处理，若发现有错误，则用事务回滚（Rollback Transaction）将此次操作所变动的资料全部

恢复。编写格式如下。

CREATE TRIGGER　触发器名称
ON 表名
AFTER 操作（INSERT、UPDATE）
AS
SQL 语句

INSTEAD OF 触发器：触发时机在资料变动前发生，且资料如何变动取决于触发器。编写格式如下。

CREATE TRIGGER 触发器名称
ON 表名
INSTEAD OF 操作（UPDATE、DELETE）
AS
SQL 语句

【问题 1】 编写触发器，实现如下功能：在订单（表 orders）中的订购数量（列名为 num）有变动时，触发器会先到客户（表 customer）中取得该用户的信用等级（列名为 level），然后再到信用额度（creit）中取出该等级许可的订购数量上下限，最后比较订单中的订购数量是否符合限制。

【解析】 触发器的代码如下。

```
CREATE  TRIGGER num_check
        ON orders
        AFTER INSERT,UPDATE
        AS
        IF UPDATE(num)
        BEGIN
            IF EXISTS(SELECT a.* FROM orders a JOIN customer b ON a.customerid=
            b.customerid
JOIN creit c ON b.level=c.level
              WHERE a.num BETWEEN c.up AND c.down)
           BEGIN
              ROLLBACK TRANSACTION
              EXEC master...xp_sendmail 'administrator','客户的订购数量不符合限制'
            END
          END
```

【问题 2】 为数据库编写触发器，实现如下功能：在工资管理系统中，当公司对某员工甲的月薪进行调整时，通常会先在员工表中修改薪资列，然后在员工记录表中修改薪资调整时间与薪资。

【解析】 触发器的代码如下。

```
CREATE TRIGGER compensation
ON 员工
     AFTER UPDATE
     AS
       IF @@rowcount=0 RETURN
       IF UPDATE(薪资)
          BEGIN
            INSERT 员工记录
            SELECT 员工编号,薪资,getdate()
            FROM inserted
          END
```

3.3　数据库应用系统的安装

数据库系统开发完毕，进入系统部署阶段。首先要综合考虑安装费用、客户关系、雇员关系、后勤关系和风险等多种因素，拟定一个合理的系统安装计划。然后按照组织机构安排的合理性原则，拟定人力资源使用计划。数据库应用系统安装方式有 3 种：直接安装、并行安装和阶段安装，本节对这 3 种方式一一做了介绍。如图 3-4 所示是本节的知识框图。

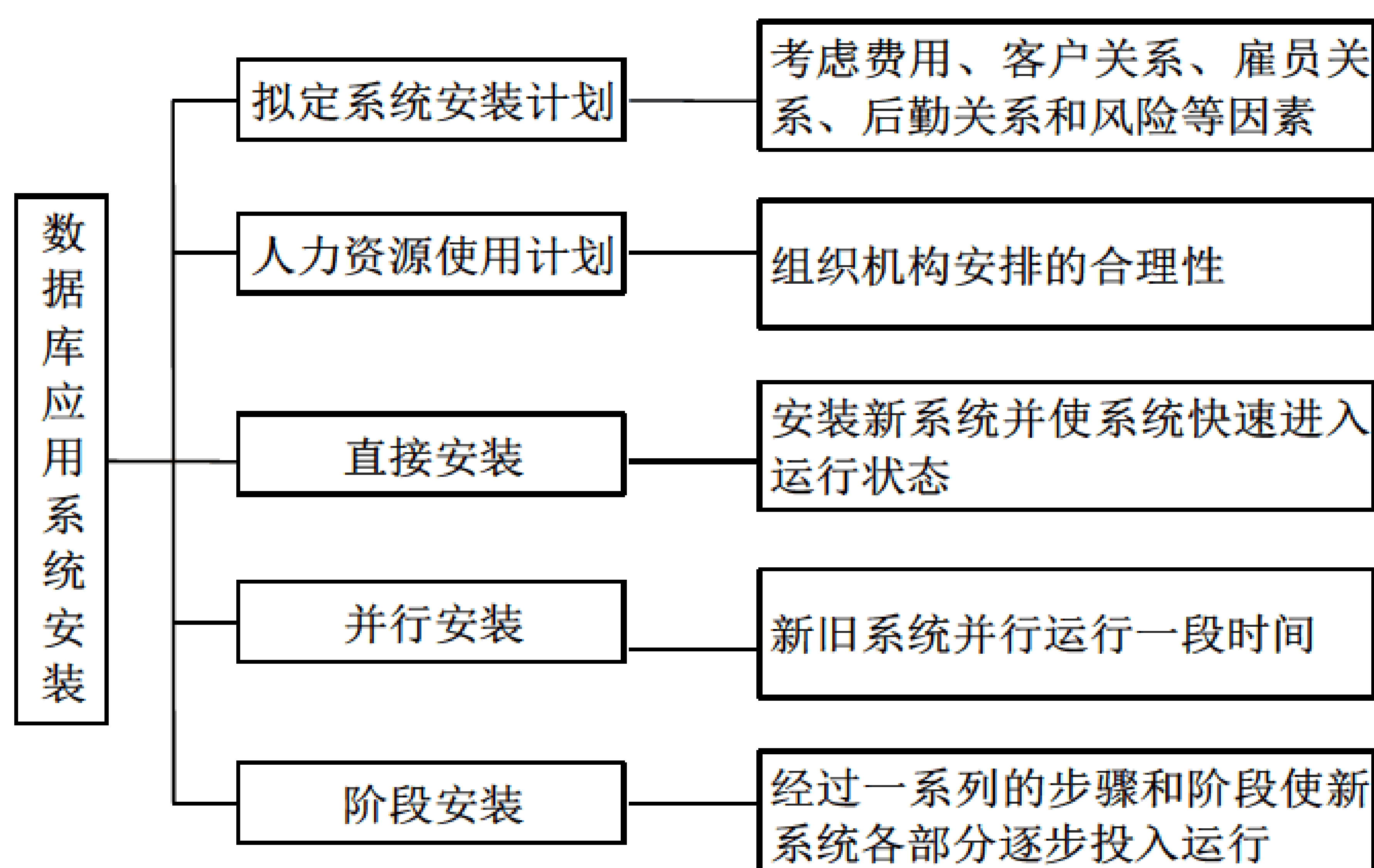


图 3-4　数据库应用系统安装知识框图

1．知识点提炼

（1）拟定系统安装计划（考虑费用、客户关系、雇员关系、后勤关系和风险等因素）

在进行新老系统转换以前，首先要进行新系统的试运行。在系统测试、调试中，这里使用的是系统测试数据，有些实际运行中可能出现的问题，很难通过这些数据被发现。所以，一个系统开发后，让它实际运行一段时间，是对系统最好的检验和测试方法。另外，

新旧系统的转换方案还要综合考虑费用、客户关系、雇员关系、后勤关系和风险等因素，拟定系统安装计划，经过评估后确实可行，则按照系统安装计划执行。

（2）拟定人力资源使用计划（组织机构安排的合理性）

在新系统取代旧系统之前，必须进行新系统的试运行，这是最好的检验和测试系统的方法。在新旧系统转换过程中，要拟定人力资源使用计划，组织机构要合理安排，保证新旧系统转换工作的顺利进行。

系统试运行阶段的主要工作有：

① 对系统进行初始化、输入各原始数据记录；

② 记录系统运行的数据和各项参数；

③ 核对新系统输出和老系统输出的结果；

④ 对新系统的输入方式进行考察，主要考虑方便程度、效率、安全可靠性、误操作保护等方面；

⑤ 对新系统实际运行的响应速度，包括运算速度、传递速度、查询速度、输出速度等，进行实际测试。

（3）直接安装（安装新系统并使系统快速进入运行状态）

如果新系统的试运行状况良好，系统的各种性能指标都达到了预定的要求，则可以考虑新系统和老系统之间的转换。新旧系统之间的转换方式有 3 种：直接转换、并行转换和分段转换。对新系统而言，其安装有直接安装、并行安装和阶段安装 3 种方式。

直接安装是指在确定新系统运行无误后，立刻启用新系统，终止旧系统的运行。这种方式对人员、设备费用、运行费用和维护费用很节省。这种转换方式适用于处理过程不太复杂、数据不很重要的系统。

（4）并行安装（新旧系统并行运行一段时间）

并行安装是指新系统与旧系统并行工作一段时间，经过一段时间的考验以后，如果新系统运行状况良好，则可以用新系统正式替代旧系统。其优点在于对于较复杂的大型系统，它提供了一个将新系统与旧系统的运行结果进行比较的机会，可以对新旧两个系统的时间要求、出错次数和工作效率给以公正的评价。它的主要特点是安全、可靠，但费用和工作量大。这种转换方式适于处理过程较为复杂、对系统的可靠性要求较高的系统。

（5）阶段安装（经过一系列的步骤和阶段使新系统各部分逐步投入运行）

阶段安装是指经过一系列的步骤和阶段使新系统各部分逐步投入运行。这种新旧系统的转换方式叫做分段转换。分段转换又称逐步转换、向导转换、试点过渡法等。这种转换方式实际上是直接安装和并行安装两种转换方式的结合。在新系统全部正式运行前，一部分一部分地代替旧系统。在转换过程中没有正式运行的部分，可以在模拟环境中继续试运行。这种安装方式的优点是既保证了可靠性，又降低了费用。但是这种分段转换要求新旧系统的各个子系统之间有一定的独立性，否则就无法采用这种阶段安装的转换方式。

在实际工作中，新旧系统转换方法较为灵活。要结合系统的实际情况，选择新旧系统

的转换方式。

2．难点分析

原有的旧系统从启用到被新系统取代，在其使用期间往往积累了大量珍贵的历史数据，其中许多历史数据都是新系统顺利启用所必需的。另外，这些历史数据也是进行决策分析的重要依据。数据迁移，就是将这些历史数据进行清洗、转换，并装载到新系统中的过程。数据迁移主要适用于一套旧系统切换到另一套新系统，或多套旧系统切换到同一套新系统时，需要将旧系统中的历史数据转换到新系统中的情况。

数据转换与迁移的过程大致可以分为抽取、转换、装载 3 个步骤。数据抽取、转换是根据新旧系统数据库的映射关系进行的，而数据差异分析是建立映射关系的前提，这其中还包括对代码数据的差异分析。转换步骤一般还要包含数据清洗的过程，数据清洗主要是针对源数据库中出现的二义性、重复、不完整、违反业务或逻辑规则等问题的数据进行的，在清洗之前需要进行数据质量分析，以找出存在问题的数据，否则数据清洗将无从谈起。数据装载是通过装载工具或自行编写的 SQL 程序将抽取、转换后的结果数据加载到目标数据库中。

数据转换是该过程的难点。例如，可以从源数据的一列中析取一个子串并将其复制到目的表。也可以使用某些特性（例如，列中的特定数据值）搜索行并只对那些行中的数据进行转换。在转变或转换异类数据和目的服务器之间的数据之前，需要考虑不同程序、提供程序以及驱动程序支持数据类型和 SQL 语句的方式的变化。一些典型的问题如下。

① 数据类型变化时，精度是否能够达到要求，例如将 real 数据类型转换为 int 数据类型。

② 字符型字段、二进制类型字段的长度是否能够满足要求，否则这些字段超出的部分将被截断。

③ 数据库管理系统的版本不一致会导致某些数据类型解释的差异。例如用于 Oracle 的 Microsoft ODBC 和 OLEDB 驱动程序支持 Oracle 7.3 BLOB 数据类型，不支持 Oracle 8.0 中的 BLOB、CLOB、NCLOB 和 BFILE 类型。

④ 数据的字符编码的转化问题，特别是汉字相关的编码问题。例如 Oracle 要求在 Unicode 字符串前加上前缀字母 N。

⑤ 数据转化错误时数据的回滚问题。

⑥ 不同数据库管理系统对 NULL 值的解释不同。

3．典型例题

【例题 3-10】 阅读以下有关项目安装的叙述，回答问题。

某单位一信息项目前端选择了 Power Designer 及 PowerBuilder 作为开发工具，采用 Client/Server 模式。在为客户建立安装包时，要求将开发的可执行文件和以下文件（以下几个文件在 PowerBuilder 的 Shared\PowerBuilder 文件夹中）一起打包，如表 3-2 所示。

【问题】 请说明这样做的原因。

表 3-2 文件及其功能

文 件 名	功 能
PBVM70.DLL	PowerBuilder 虚拟机
PBTRA60.DLL	用于数据库跟踪调用
PBRTC60.DLL	对 Rich Text 的支持
PBMSS70.DLL	Microsoft SQL Server 数据库服务器的直连接口（Native database interfaces），如果使用别的 DBMS，有相应的其他 DLL
PBDWE60.DLL	DataWindow 引擎，支持 DataWindow 和 Report
NTWDBLIB.DLL	DBMS 客户端连接库，负责执行与服务器的连接
DBNMPNTW.DLL	Named Pipes Network Library，网络连接方式——命名管道
DBMSSOCN.DLL	TCP/IP Network Library，网络连接方式——TCP/IP
PBODB70.DLL，PBODB70.IN	支持 ODBC 数据库接口

【解析】 经过编译生成的 PowerBuilder 应用程序需要一定的运行环境。一个 EXE 文件（或者再加 PBD 文件）要提交给脱离了 PowerBuilder 环境的用户使用时，还必须提供一些 PowerBuilder 应用程序执行、数据库连接等实现所必需的环境动态连接库文件。如果缺少这些 DLL 文件，应用程序可能无法启动，或者无法连接到数据库服务器。

【例题 3-11】 阅读以下有关安装测试的叙述，回答问题。

安装测试是软件测试的一个重要组成部分。安装测试的目的不是找软件错误，而是找安装错误。在安装软件系统时，会有多种选择。

① 要分配和装入文件与程序库。

② 布置适用的硬件配置。

③ 进行程序的连接。

而安装测试就是要找出在这些安装过程中出现的错误。安装测试是在系统安装之后进行测试。试列举它要检验哪些方面。

【解析】 要检验以下几个方面：

① 用户选择的一套任选方案是否相容；

② 系统的每一部分是否都齐全；

③ 所有文件是否都已产生并确有所需要的内容；

④ 硬件的配置是否合理，等等。

【例题 3-12】 阅读以下有关安装测试的叙述，回答问题。

除了嵌入式软件之外，安装是软件产品实现其功能的第一步，没有正确的安装根本就谈不上正确的执行，因此对于安装的测试就显得尤为重要。

某单位一信息项目的数据库采用了 Oracle 7.3 作为后端平台，前端选择了 Oracle 公司的 Developer 2000 及 Designer 2000 作为开发工具，采用 Client/Server 模式。关于该应用系

统安装测试需要注意的问题提出了如下一些想法。

① 自动安装还是手工配置安装，测试各种不同的安装组合，并验证各种不同组合的正确性，最终目标是所有组合都能安装成功。

② 安装退出之后，确认应用程序可以正确启动、运行。

③ 在安装之前应备份注册表，安装之后，查看注册表中是否有多余的垃圾信息。

④ 卸载测试和安装测试同样重要，如果系统提供自动卸载工具，那么卸载之后需检验系统是否把所有的文件全部删除，注册表中有关的注册信息是否也被删除。

⑤ 至少要在一台笔记本电脑上进行安装测试，因为有很多产品在笔记本电脑中会出现问题，尤其是最终用户级的产品。

⑥ 安装完成之后 ，可以在简单地使用之后再执行卸载操作，有的系统在使用之后会发生变化，变得不可卸载。

⑦ 对于 Client/Server 模式的应用系统，应该先安装服务器端，然后安装客户端，测试是否会出现问题。

⑧ 考察安装该系统是否对其他的应用程序造成影响，特别是 Windows 操作系统，经常会出现此类的问题。

【问题】 在上述 8 条叙述中有 2 条是不正确或不恰当的，请指出其序号，并简要说明理由（50 字以内）。

【解析】 第 5 条，至少要在一台笔记本电脑上进行安装测试，因为有很多产品在笔记本电脑中会出现问题，尤其是系统级的产品。一般来说，笔记本电脑和台式机存在硬件上的差异，跟底层系统相关的软件，如驱动程序等和台式机差异较大，而上层应用的差异较小。

第 7 条，对于 Client/Server 模式的应用系统，可以先安装客户端，然后安装服务器端，测试是否会出现问题，因为很多客户端的问题都是没有考虑到服务器不存在或连接不成功的情况。当然，服务器安装成功后，还需要对服务器连接等功能进行再测试。

3.4　数据库应用系统的测试

本节主要介绍数据库应用系统的测试，涵盖拟定测试目标、计划、方法与步骤，数据加载，准备测试数据，指导应用程序员进行模块测试、验收，准备系统集成、测试环境、测试工具，写出数据库运行测试报告等知识点。如图 3-5 所示是本节的知识框图。

1．知识点提炼

（1）拟定测试目标、计划、方法与步骤

系统测试是为了发现错误而执行程序的过程。测试的目的就是希望能以最少的人力和时间发现潜在的各种错误和缺陷。应根据开发各阶段的需求、设计等文档或程序的内部结构精心设计测试实例，并利用这些实例来运行程序，以便发现错误。



图 3-5　数据库应用系统的测试知识框图

为了确保应用系统的质量，测试的准备工作应该在分析和设计阶段就开始了。当设计工作完成以后，就应该着手测试的准备工作。一般来讲，由对整个系统设计熟悉的设计人员编写测试大纲，明确测试的内容和测试通过的准则，设计完整合理的测试用例，以便系统实现后进行全面测试。

（2）程序的测试

软件测试。是整个软件开发过程中交付用户使用前的最后阶段，是软件质量保证的关键。软件测试在软件生存周期中横跨两个阶段：通常在编写出每一个模块之后，就对它进行必要的测试（称为单元测试）。编码与单元测试属于软件生存周期中的同一阶段。该阶段的测试工作就是编程组内部人员进行的交叉测试（避免编程人员测试自己的程序）。这一阶段结束后，进入软件生存周期的测试阶段，对软件系统进行各种综合测试。测试工作由专门的测试组完成，测试组设组长一名，负责整个测试的计划、组织工作。测试组的其他成员由具有一定分析、设计和编程经验的专业人员组成，人数根据具体情况可多可少，一般 3～5 人为宜。

在系统的实现组将所开发的程序经过初步的验证后，提交系统测试组，由测试负责人组织测试工作。

测试应该贯穿于系统定义与开发的整个过程。因此，对分析、设计和实现等各阶段所得到的结果，包括需求规格说明、设计规格说明及源程序都应进行测试。测试准备过程需要做好以下工作。

① 系统的设计和实现都是基于需求分析规格说明进行的，需求分析规格说明是否完整、正确、清晰是系统开发成败的关键。为了保证需求定义的质量，应对其进行严格的审查。仔细阅读有关资料，包括系统的规格说明书、设计文档、使用说明书以及在系统设计

过程中编写的测试大纲、测试内容、测试的通过准则等，全面熟悉整个系统。

② 设计评审。系统设计是将系统需求转换成软件表示的过程，主要描绘出系统结构、详细的处理过程和数据库模式。按照需求的规格说明对系统结构的合理性、处理过程的正确性进行评价，同时利用关系数据库的规范化理论对数据库模式进行审查。

③ 编写测试计划，设计测试用例，做好测试前的准备工作。

为了保证测试的质量，一般将测试过程分成代码审查、单元测试、集成测试和验收测试几个阶段，具体如下。

- 代码会审。代码会审是由会审小组通过阅读和讨论对程序进行静态分析的过程。会审小组在充分阅读待审程序文本、控制流程图、有关要求及规范等文档的基础上，由程序员讲解程序的逻辑，并展开讨论，以揭示错误的关键所在。
- 单元测试。单元测试集中在检查软件设计的最小单位——模块上，通过测试检查该模块的实际功能与定义该模块的功能说明是否相符合，以及编码是否存在错误。高可靠性的模块是组成可靠系统的坚实基础。
- 集成测试。集成测试是将模块按照设计要求组装起来同时进行测试，主要目标是发现与接口有关的问题。比如数据穿过接口时可能丢失，子功能组合起来可能没有实现预期的主功能，全程数据结构可能有错误等。
- 验收测试。验收测试的目的是向未来的用户表明系统能够像预定要求那样工作。经集成测试后，已经按照设计把所有的模块组装成一个完整的软件系统，接口错误也已经基本排除了，接着就应该进一步验证软件的有效性，这就是验收测试的任务，即软件的功能和性能是否如用户所期待的那样。

按照上述的测试过程对数据库应用系统进行测试后，系统基本满足开发的要求，测试完毕，经验收后，将系统提交用户运行。

（3）数据加载，准备测试数据

完成数据库的设计和应用程序的设计之后，开发人员根据设计，用选定的 RDBMS 提供的 SQL 语言的数据定义语言（DDL）及其他高级语言对设计进行代码编写，经过调试运行后，就建立了系统数据库的结构，包括数据库及基本表、索引、约束等数据库对象也都建立起来了。接下来就要加载实验数据，进行数据库系统的试运行。实验数据不完全等同于实际运行中的数据。经过试运行后，如果系统的各项功能都已经实现，并且系统性能达到预定的要求，就可以卸载实验数据，加载用户数据，使系统正式运行。如果用户数据是以前旧系统的数据，不一定能够完全满足新系统的数据要求，需要对其进行处理，同时还要做好新系统数据库的转储和恢复工作，以免发生故障时丢失数据。

（4）指导应用程序员进行模块测试、验收

模块测试由于模块规模小、功能单一、逻辑简单，测试人员可以通过模块说明书和源程序，清楚地了解该模块的 I/O 条件和模块的逻辑结构，采用结构测试（白盒法）的用例，尽可能达到彻底测试，然后辅之以功能测试（黑盒法）的用例，使之对任何合理和不合理

的输入都能鉴别并响应。进行单元测试在为模块设计程序用例时，可以直接参考模块的源程序。所以单元测试的策略是综合运用白盒法和黑盒法。具体做法有两种。

第一种方法是先用黑盒法提出一组基本的测试用例，然后用白盒法验证。如果发现用黑盒法产生的测试用例未能满足所需的覆盖标准，就用白盒法增补新的测试用例来满足它们。覆盖的标准应该根据模块的具体情况确定。对可靠性要求较高的模块，通常要满足条件组合覆盖或路径覆盖标准。

第二种方法是先用白盒法分析模块的逻辑结构，提出一批测试用例，然后根据模块的功能用黑盒法进行补充。

集成测试及其后的测试阶段，一般采用黑盒法，其策略包括：

① 用边值分析法或等价分类法提出基本的测试用例；

② 用猜测法补充新的测试用例。

如果在程序的功能说明中含有输入条件的组合，应该首先用因果图法，然后再按照上面所述的步骤①和②进行。

验收测试的目的是向未来的用户表明系统能够像预定要求那样工作。经集成测试后，已经按照设计把所有的模块组装成一个完整的软件系统，接口错误也已经基本排除了，接着就应该进一步验证软件的有效性，这就是验收测试的任务，即软件的功能和性能如同用户所期待的那样。

（5）写出数据库运行测试报告

系统测试文件描述要执行的软件测试及测试的结果，测试文件对于保证系统的质量和系统的运行有着重要意义。测试时必须把系统测试要求、过程以及测试结果以正式的文件形式写出。

根据测试文件所起的作用不同，通常把测试文件分成两类，即测试计划和测试分析报告。

测试计划详细规定测试的要求，包括测试的目的、内容、方法、步骤、测试的准则等。由于要测试的内容可能涉及到软件的需求和软件的设计，因此必须及早开始测试计划的编写工作。不应在着手测试时，才开始考虑测试计划。通常，测试计划的编写从需求分析阶段开始，到软件设计阶段结束时完成。

测试报告用来对测试结果进行分析说明，经过测试后，证实了软件具有的能力，以及它的缺陷和限制，并给出评价的结论性意见，这些意见既是对软件质量的评价，又是决定该软件能否交付用户使用的依据。由于要反映测试工作的情况，自然要在测试阶段内编写。

2．难点分析

在实际的数据库项目中，如何测试数据库应用，测试内容有哪些？这些问题都是数据库开发人员经常遇到的难题。《国家计算机标准和文件模板》中给出的《测试分析报告》，能帮助解决这些问题。

现摘录文档模板如下。

1　引言

1.1　编写目的

说明这份测试分析报告的具体编写目的，指出预期的阅读范围。

1.2　背景

说明：

a．被测试软件系统的名称；

b．该软件的任务提出者、开发者、用户及安装此软件的计算中心，指出测试环境与实际运行环境之间可能存在的差异以及这些差异对测试结果的影响。

1.3　定义

列出本文件中用到的专业术语的定义和外文首字母组词的原词组。

1.4　参考资料

列出要用到的参考资料，如：

a．本项目经核准的计划任务书或合同、上级机关的批文；

b．属于本项目的其他已发表的文件；

c．本文件中各处引用的文件、资料，包括所要用到的软件开发标准。列出这些文件的标题、文件编号、发表日期和出版单位，说明这些文件资料的来源。

2　测试概要

用表格的形式列出每一项测试的标识符及其测试内容，并指明实际进行的测试工作内容与测试计划中预先设计的内容之间的差别，说明做出这种改变的原因。

3　测试结果及发现

3.1　测试 1（标识符）

把本项测试中实际得到的动态输出（包括内部生成数据输出）结果同对于动态输出的要求进行比较，陈述其中的各项发现。

3.2　测试 2（标识符）

用类似本报告 3.1 条的方式给出第 2 项及其后各项测试内容的测试结果和发现。

4　对软件功能的结论

4.1　功能 1（标识符）

4.1.1　能力

简述该项功能，说明为满足此项功能而设计的软件能力以及经过一项或多项测试已证实的能力。

4.1.2　限制

说明测试数据值的范围（包括动态数据和静态数据），列出就这项功能而言，测试期间在该软件中查出的缺陷、局限性。

4.2　功能 2（标识符）

用类似本报告 4.1 的方式给出第 2 项及其后各项功能的测试结论。

5　分析摘要

5.1　能力

陈述经测试证实了的本软件的能力。如果所进行的测试是为了验证一项或几项特定性能要求的实现，应提供这方面的测试结果与要求之间的比较，并确定测试环境与实际运行环境之间可能存在的差异对能力的测试所带来的影响。

5.2　缺陷和限制

陈述经测试证实的软件缺陷和限制，说明每项缺陷和限制对软件性能的影响，并说明全部测得的性能缺陷的累积影响和总影响。

5.3　建议

对每项缺陷提出改进建议，如：

a．各项修改可采用的修改方法；

b．各项修改的紧迫程度；

c．各项修改预计的工作量；

d．各项修改的负责人。

5.4　评价

说明该项软件的开发是否已达到预定目标，能否交付使用。

6　测试资源消耗

总结测试工作的资源消耗数据，如工作人员的水平、级别、数量、机时消耗等。

3．典型例题

【例题 3-13】 黑盒测试方法是在程序接口上进行测试，主要是为了发现哪几种类型的错误？

【解析】 黑盒测试方法主要是为了发现如下几种类型的错误:

① 是否有不正确或遗漏了的功能；

② 在接口上，输入能否正确地接收；能否输出正确的结果；

③ 是否有数据结构错误或外部信息（例如数据文件）访问错误；

④ 性能上是否能够满足要求；

⑤ 是否有初始化或终止性错误。

【例题 3-14】 软件测试一般有哪些原则？

【解析】 一般来说，软件测试遵循如下原则：

① 应当尽早和不断地进行软件测试；

② 测试用例应由测试输入数据和预期输出结果两部分组成（注意测试用例包含输入和输出两部分）；

③ 测试用例应包括合理的输入条件和不合理的输入条件，前者即程序正常运行的条件，后者即可能引起程序非法操作的条件；

④ 程序员应避免检查自己的程序，程序员总是在潜意识里不愿意否定自己的劳动成果；

⑤ 注意测试中的群集现象，测试后程序中残存的错误数目与该程序中已发现的错误数目成正比，所以不要在某个程序段中找到几个错误就误认为该程序段就没有错误而不再测试，相反该程序段更要集中精力测试；

⑥ 测试应该制定计划，严格按计划进行测试，避免测试的随意性，应对每一个测试结果做全面检查，有时出错的征兆已经在测试结果中出现了，但由于没有对测试结果进行仔细检查，而使这个错误成了漏网之鱼；

⑦ 应妥善保存测试计划、用例、错误记录和分析报告。

【例题 3-15】 对一个具有多重选择和循环嵌套的程序，不同的路径数目可能是天文数字。如图 3-6 所示是一个小程序的流程图，它包括了一个执行 20 次的循环。试计算所有可能的测试路径。如果对每一条路径进行测试需要 1 毫秒，试计算总共的测试时间。

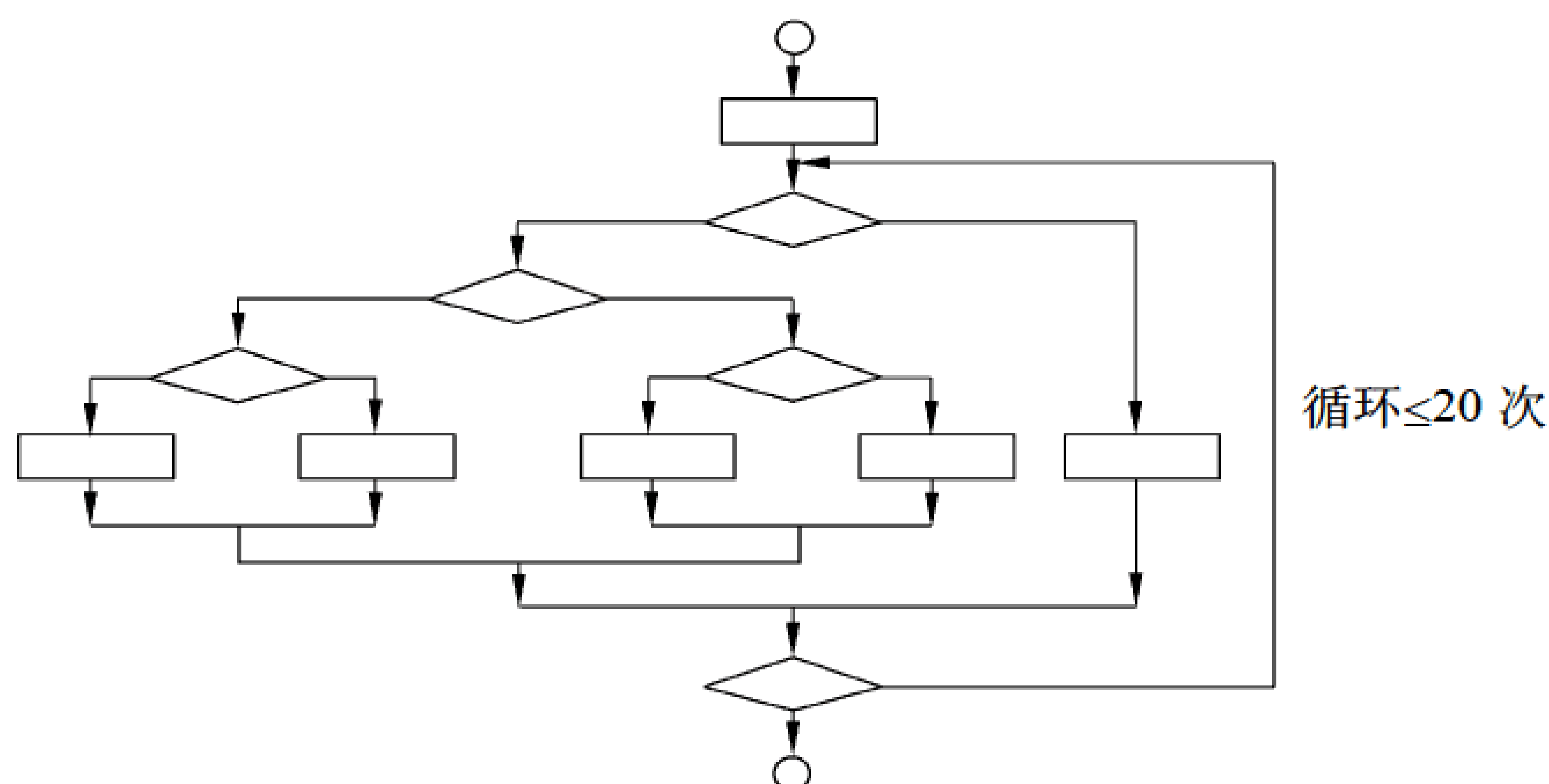


图 3-6　小程序流程图

【解析】 该程序包含的不同执行路径数达 520 条，对每一条路径进行测试需要 1 毫秒，假定一年工作 365 × 24 小时，要想把所有路径测试完，需 3170 年。

【例题 3-16】 为把握软件开发各个环节的正确性，需要进行各种确认和验证工作。请简述确认和验证工作的意义。

【解析】 确认（Validation），是一系列的活动和过程，目的是想证实在一个给定的外部环境中软件的逻辑正确性。例如：需求规格说明的确认、程序的确认（静态确认、动态确认）。

验证（Verification），试图证明在软件生存期各个阶段，以及阶段间的逻辑协调性、完备性和正确性。

【例题 3-17】 在单元测试时，如果程序有 I/O 操作，测试人员应重点考虑哪些问题？

【解析】 测试人员应重点考虑：

① 文件属性是否正确；

② OPEN 与 CLOSE 语句是否正确；

③ 缓冲区容量与记录长度是否匹配；

④ 在进行读写操作之前是否打开了文件；

⑤ 在结束文件处理时是否关闭了文件；

⑥ 正文书写、输入错误；

⑦ I/O 错误是否检查并做了处理。

【例题 3-18】 软件的测试过程有 4 个步骤：单元测试、集成测试、确认测试和系统测试。根据你自己的项目经验，谈谈什么是单元测试、集成测试、确认测试和系统测试？

【解析】 单元测试即对每一个单元模块进行测试。然后把测试过的模块组装起来进行集成测试，主要是对软件体系结构的构造进行测试。接着进行确认测试，检查软件是否满足了各种需求，以及配置是否合理安全。最后是系统测试，即把经确认测试后的软件放到实际运行环境中，与系统的其他构件一起进行测试。

【例题 3-19】 单元测试时，有时需要为测试的模块编写辅助模块：驱动模块和桩模块，它们的区别是什么？

【解析】 驱动模块是用来调用被测模块的；桩模块用来代替被测模块调用的子模块。

【例题 3-20】 阅读以下有关软件测试的叙述，回答问题。

等价类划分是一种典型的黑盒测试方法，使用这一方法时，完全不考虑程序的内部结构，只依据程序的规格说明来设计测试用例。

等价类划分方法把所有可能的输入数据，即程序的输入域划分成若干部分，然后从每一部分中选取少数有代表性的数据作为测试用例。等价类是指某个输入域的子集合。在该子集合中，各个输入数据对于揭露程序中的错误都是等效的。测试某等价类的代表值就等价于对这一类其他值的测试。等价类的划分有两种不同的情况。

① 有效等价类是指就程序的规格说明而言，是合理的、有意义的输入数据构成的集合。

② 无效等价类是指就程序的规格说明而言，是不合理的、无意义的输入数据构成的集合。

在设计测试用例时，要同时考虑有效等价类和无效等价类的设计。

【问题】 在某一 Pascal 语言版本中规定：“标识符是由字母开头，后跟字母或数字的任意组合构成的。有效字符数为 8 个，最大字符数为 80 个。”并且规定：“标识符必须先说明，再使用。在同一说明语句中，标识符至少有一个。”用等价类划分法设计测试用例。

【解析】 用等价类划分的方法，建立输入等价类表（具体的测试用例略），如表 3-3

所示。

表 3-3　输入等价类表

输 入 条 件	有效等价类	无效等价类
标识符个数	1 个（1），多个（2）	0 个（3）
标识符字符数	1～8 个（4）	0 个（5），>8 个（6），>80 个（7）
标识符组成	字母（8），数字（4）	非字母数字字符（10），保留字（11）
第一个字符	字母（12）	非字母（13）
标识符使用	先说明后使用	未说明已使用（15）

【例题 3-21】 为了判断应用系统是否合格，用预先确定的一系列数据在系统中运行，并与预期的结果进行比较，这一过程称为测试。它是软件质量保证的重要手段。有人认为测试和调试是一回事，那么测试和调试可以相互替代吗?

【解析】 简单地说，测试是一种检验，经过测试人们会看到一些现象。这些现象也许是可疑的征兆，但往往不能直接从测试结果中找到错误的根源。这就需要充分利用测试结果和测试提供的信息进行全面分析，以便找到错误的根源和出现错误的原因。紧接着便是纠正已发现的错误。测试以后进行的这些工作称为调试或排错。

我们不能把两者混为一谈。但它们毕竟有着密切的关系，常常是在测试以后紧接着就要着手排错。实际上，这两种工作经常交叉进行，是不可相互替代的。

【例题 3-22】 阅读以下有关软件测试的叙述，回答问题 1 和问题 2。

如图 3-7 所示是测试阶段的信息流图。

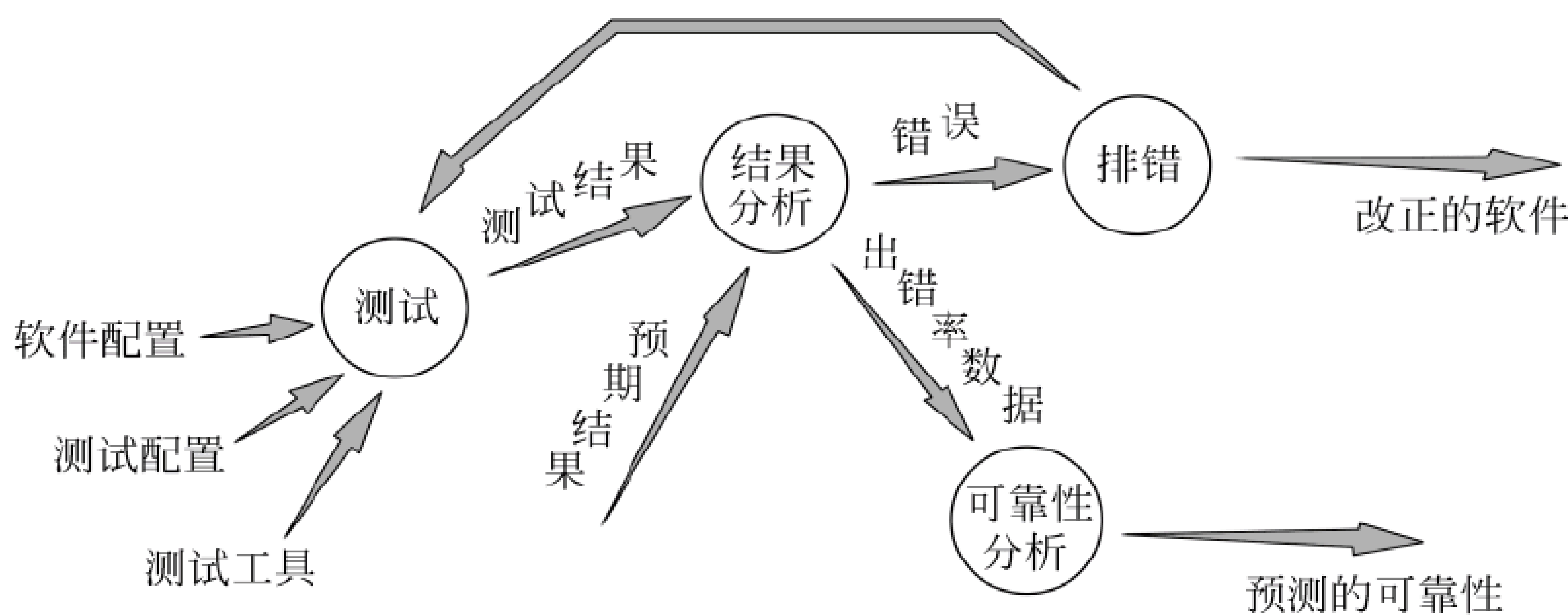


图 3-7　测试阶段的信息流图

【问题 1】 图中软件配置、测试配置、测试工具主要指什么？

【解析】

软件配置：软件需求规格说明、软件设计规格说明、源代码等；

测试配置：测试计划、测试用例、测试程序等；

测试工具：测试数据自动生成程序、静态分析程序、动态分析程序、测试结果分析程

序、驱动测试的测试数据库等等。

【问题 2】 在图 3-7 中，排错（调试）主要包含哪些工作？

【解析】 排错（调试）主要包含：对已经发现的错误进行错误定位和确定出错性质，并改正这些错误，同时修改相关的文档。

【例题 3-23】 通常，在单元测试的基础上，需要将所有模块按照设计要求组装成为系统，相对应的是组装测试（集成测试、联合测试）。阅读以下有关组装测试的叙述，回答问题。

在单元测试的同时可进行组装测试，发现并排除在模块连接中可能出现的问题，最终构成要求的软件系统。这时需要考虑的问题主要有哪些？

【解析】 这时需要考虑的问题主要有如下几点：

① 在把各个模块连接起来的时候，穿越模块接口的数据是否会丢失；

② 一个模块的功能是否会对另一个模块的功能产生不利影响；

③ 各个子功能组合起来，能否达到预期要求的父功能；

④ 全局数据结构是否有问题；

⑤ 单个模块的误差累积起来，是否会放大，从而达到不能接受的程度。

【例题 3-24】 阅读以下有关组装测试的叙述，回答问题。

通常，把模块组装成为系统的方式有两种：一次性组装方式和增殖式组装方式。

一次性组装方式（big bang）是一种非增殖式组装方式，也叫做整体拼装。使用这种方式，首先对每个模块分别进行模块测试，然后再把所有模块组装在一起进行测试，最终得到要求的软件系统。

增殖式组装方式又称渐增式组装。首先对一个个模块进行模块测试，然后将这些模块逐步组装成较大的系统，在组装的过程中边连接边测试，以发现连接过程中产生的问题，通过增殖逐步组装成为要求的软件系统。

【问题】 增殖式组装方式又分为自顶向下的增殖方式和自底向上的增殖方式两种。如图 3-8 所示是自顶向下的增殖方式。

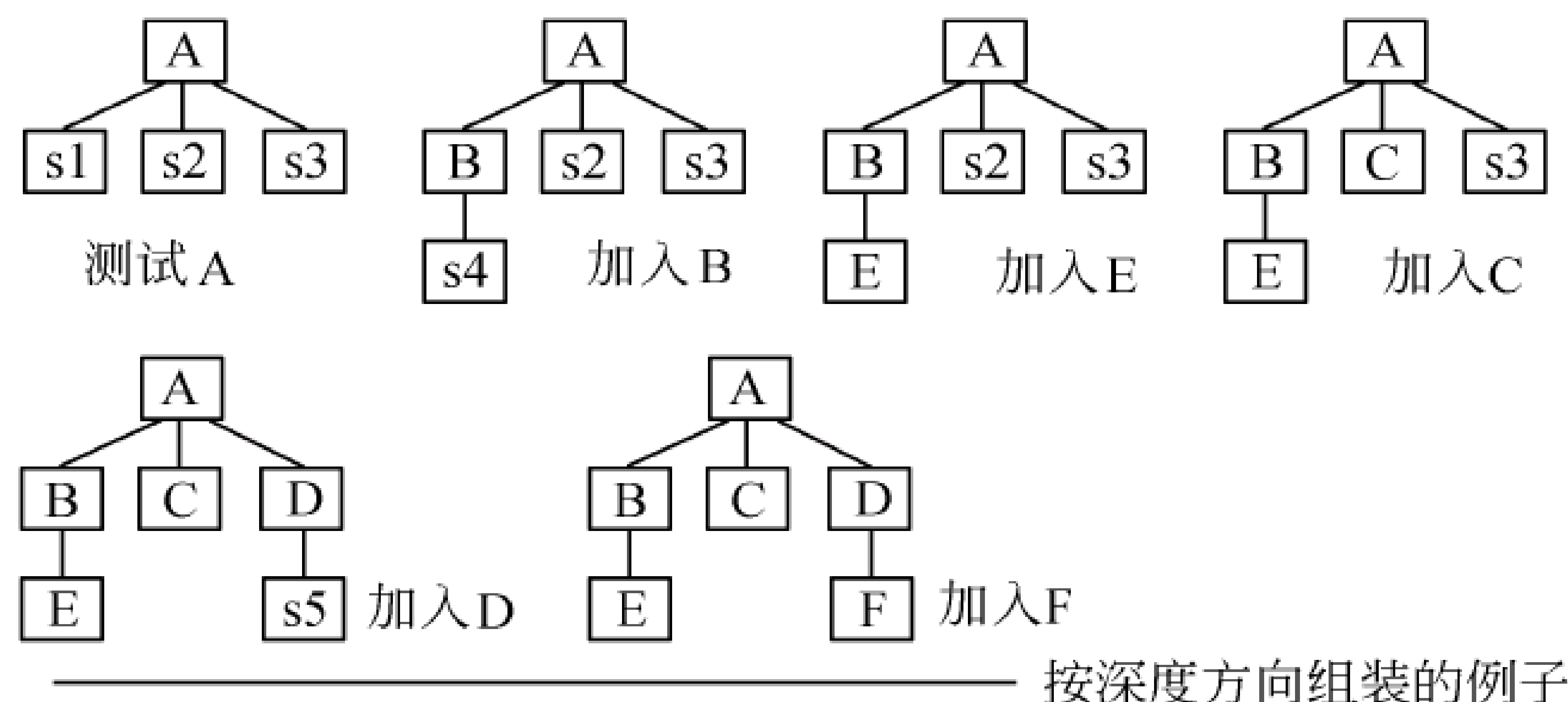


图 3-8　自顶向下的增殖方式

如图 3-9 所示是自底向上的增殖方式。

请比较这两种方式的优缺点。

【解析】 自顶向下增殖的方式和自底向上增殖的方式各有优缺点。一般来讲，一种方式的优点是另一种方式的缺点。

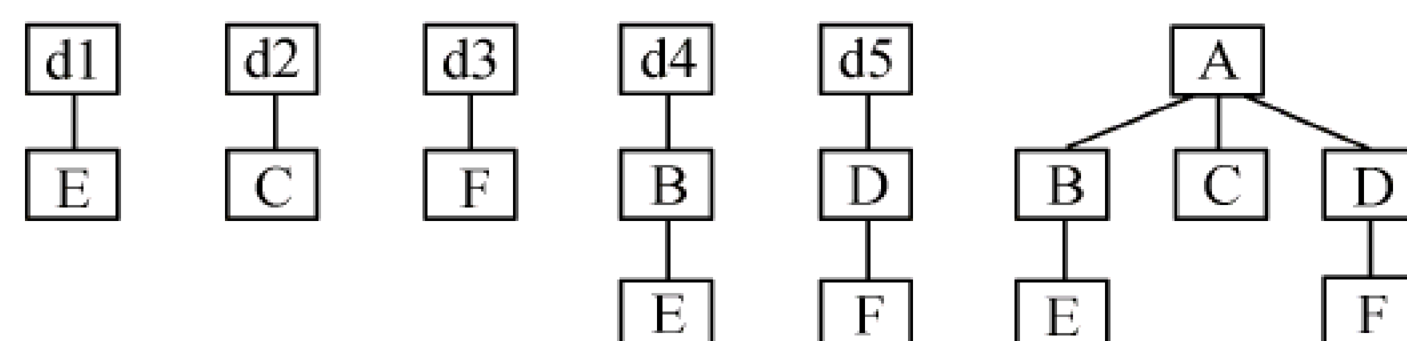


图 3-9　自底向上的增殖方式

自顶向下方式将模块按系统程序结构，沿控制层次自顶向下进行组装。自顶向下的增殖方式在测试过程中较早地验证了主要的控制和判断点。选用按深度方向组装的方式，可以通过首先实现和验证一个完整的软件功能。

自底向上的方式是从程序模块结构的最底层模块开始组装和测试的。因为模块自底向上进行组装，对于一个给定层次的模块，它的子模块（包括子模块的所有下属模块）已经组装并测试完成，所以不再需要桩模块。在模块的测试过程中需要从子模块得到的信息可以通过直接运行子模块得到。

【例题 3-25】 阅读以下有关性能测试的叙述，回答问题。

在某市话费管理系统中，每月 20 日左右是市话交费的高峰期，全市几千个收费网点同时启动。收费过程一般分为两步，首先要根据用户提出的电话号码来查询出其当月的费用，然后收取现金并将此用户修改为已交费状态。一个用户的操作看起来很简单，但当成百上千的终端同时执行这样的操作，情况就大不一样了，如此众多的交易同时发生，对应用程序本身、操作系统、中心数据库服务器、中间件服务器、网络设备的承受力都是一个严峻的考验。

【问题】 负责该系统的王经理认为：不可能在发生问题后才考虑系统的承受力，预见软件的并发承受力是在软件测试阶段就应该解决的问题。如何在性能测试中模拟上千个并发用户的实际情况呢？请帮助王经理给出测试基本策略。

【解析】 测试的基本策略是自动负载测试，通过在一台或几台 PC 上模拟成百或上千的虚拟用户同时执行业务的情景，对应用程序进行测试，同时记录下每一事务处理的时间、中间件服务器峰值数据、数据库状态等。通过可重复的、真实的测试能够彻底度量应用的可扩展性和性能，确定问题所在以及优化系统性能。预先知道了系统的承受力，就为最终用户规划整个运行环境的配置提供了有力的依据。

【例题 3-26】 Rational 的测试套件中包括团队功能测试工具（Rational TeamTest）、涵盖测试范围评估工具（Rational Visual PureCoverage）、视觉化功能测试工具（Rational Visual Test）、软件纠错工具（Rational Purify）等等。请说明其基本用途。

【解析】 团队功能测试工具（Rational TeamTest）能够提供质量工程师执行全方位的软件测试，包括测试规划和准备、自动产生测试代码（Test Script）、自动测试执行、瑕疵追踪（Defect Tracking）与更正、测试报告、测试结果制图以及测试进度评估等重要功能及特色。

涵盖测试范围评估工具（Rational Visual PureCoverage）针对 C/C++、Visual Basic 或 Java 程序，使程序员或软件测试工程师彻底地找出被测应用程序的涵盖测试统计信息，以

利于辨别并指出尚未测试的程序代码。

视觉化功能测试工具(Rational Visual Test)针对 C/C++、Visual Basic、HTML 或 Dynamic HTML 程序，让程序员或软件测试人员快速利用图形化界面产生测试脚本，以进行自动化回归测试，进而缩短测试时间。

软件纠错工具（Rational Purify）针对 C/C++程序，使程序员在程序执行期间纠正各类错误，例如内存错误等。

【例题 3-27】 阅读以下有关并发性能测试的叙述，回答问题 1 和问题 2。

并发性能测试的过程是一个负载测试和压力测试的过程，即逐渐增加负载，直到系统的瓶颈或者不能接受的性能点，通过综合分析交易执行指标和资源监控指标来确定系统并发性能的过程。

【问题 1】 负载测试和压力测试的区别和联系是什么？

【解析】 负载测试（Load Testing）是确定在各种工作负载下系统的性能，目标是测试当负载逐渐增加时，系统组成部分的相应输出项，例如通过量、响应时间、CPU 负载、内存使用等来决定系统的性能。负载测试是一个分析软件应用程序和支撑架构、模拟真实环境的使用，从而来确定能够接收的性能过程。压力测试（Stress Testing）是通过确定一个系统的瓶颈或者不能接受的性能点，来获得系统能提供的最大服务级别的测试。

【问题 2】 并发性能测试的目的是什么？

【解析】 并发性能测试的目的主要体现在 3 个方面：以真实的业务为依据，选择有代表性的、关键的业务操作设计测试案例，以评价系统的当前性能；当扩展应用程序的功能或者新的应用程序将要被部署时，负载测试会帮助确定系统是否还能够处理期望的用户负载，以预测系统的未来性能；通过模拟成百上千个用户，重复执行和运行测试，可以确认性能瓶颈并优化和调整应用，目的在于寻找到瓶颈问题。

【例题 3-28】 阅读以下有关单元测试的叙述，回答问题 1 和问题 2。

单元测试是在软件开发过程中要进行的最低级别的测试活动，在单元测试活动中，软件的独立单元将在与程序的其他部分相隔离的情况下进行测试。

在一种传统的结构化编程语言中，比如 C，要进行测试的单元一般是函数或子过程。在像 C++这样的面向对象的语言中，要进行测试的基本单元是类。

【问题 1】 在某数据库应用中，技术员小王要进行白盒测试如下一个类的成员函数：int mode2（int nParam）。马工程师建议他生成如下 6 个测试用例：nParam = 1，0，–1，2 147 483 647，–2 147 483 647，230。马工程师生成测试用例的原则是什么？

【解析】 软件测试用例的生成主要还是测试一些边界值，例如最大值、最小值、0 等。

【问题 2】 假设要测试如下一个类的成员函数：void strcpy(char* dest, char const * src)，请给出 9 个测试用例。

【解析】

① dest = NULL, src = NULL

② dest = "yPqKIJ!u_", src = NULL

③ dest = "", src = NULL

④ dest = NULL, src = "h)zn9b"

⑤ dest = "BsmC,/i=zI6CT}pX", src = "HcI{BeP(J"

⑥ dest = "", src = "% i?～TnON"

⑦ dest = NULL, src = ""

⑧ dest = "($MN<n;^", src = ""

⑨ dest = "", src = ""

【例题 3-29】 阅读以下关于软件可靠性需求分析方面的叙述，回答问题 1 到问题 3。

某企业信息部门的李工程师正在为其下属单位开发一个应用软件，在编写软件需求规格说明书时，涉及到如何定量地描述软件可靠性的问题。

李工认为软件可靠性指的是在将要使用的指定环境下，软件能以用户可接受的方式正确运行任务所表现出来的能力。从定量角度看，似乎应当是该软件在约定的环境条件下和在给定的时间区间内，按照软件规格说明的要求，成功地运行程序所规定功能的概率。但是，他感到要具体地定量描述有些困难。

为此，李工查阅到了本部门某个软件需求规格说明书中有关的一段内容：

① 在集成与系统测试期间，由非开发组人员参与测试，每 10 千行可执行代码可能检测到的错误（bug）不能大于 6 个；

② 在提交使用的系统中，每 10 千行可执行代码可能保留着的错误数不能大于 8 个；

③ 在第一年工作期间，系统在 99.9%的工作日内，应能保持 100%的正常工作状态。

在上述说明后，还有一条注解是：错误（bug）可采用蒙特卡罗（MonteCarlo）随机植入技术进行测试。

李工程师首先想到了曾经学到过采用蒙特卡罗随机统计技术确定不规则形状封闭图形面积的方法，即是采用一个大的矩形把待测的封闭图形完全包围在该大矩形的内部，由计算机大量生成在此矩形内均匀分布的“点”，然后清点一下在大矩形内总的“点”数和在封闭图形内的“点”数，应当近似有：

$$\text{封闭图形的面积}=\frac{\text{在封闭图形内的点的个数}\times\text{已知的大矩形的面积}}{\text{大矩形内总的点的个数}}$$

【问题 1】 如果把这个思想应用于系统测试过程，先在某个程序中随机地人为植入 10 个错误，然后由一个测试组进行测试，结果一共发现有 120 个错误，其中有 6 个是人为植入的错误。

请估算一下这时该程序中会遗留下多少个未被发现的隐藏错误。同时请在 100 字以内简要地以提纲方式列举出采用这种错误随机植入方式来估算系统中遗留错误所固有的局限性。

【解析】 程序总错误数≈10 × 120/6 = 200 个，遗留未能发现的错误数= 200 – 120 – 4 = 76 个。采用这种错误随机植入方式来估算系统中遗留错误所固有的局限性有以下几点：

① 所有错误不会平等地出现；

② 错误有连带相关性（一个错误的存在可能潜藏有另一个错误）；

③ 测试检测错误时，错误不是等同可发现的；

④ 修复错误常会引起新的错误。

【问题 2】 在进行上述分析后，李工程师感到有些困惑，于是与本企业维护系统的一位系统管理员进行了讨论，系统管理员告诉他可以借用硬件的 MTTF（失效的平均等待时间，MeanTime To Failure）或者 MTBF（失效的平均间隔时间）作为软件可靠性的主要指标。

这时，李工程师查到了本企业中的一个典型例子：某软件在提交使用后，在第 1 周内有 5 次软件故障（查出了有关的 bug），在第 2 周至第 4 周内共有 2～3 次出错（也排除了错误根源），在 2 个月以后该软件一直能正常使用运行（大家反应不错），一直到 6 年半后的一天突然停工，即工作不正常。

请在 100 字以内分析该软件最后一次工作不正常的可能原因，并说明 MTBF 是在什么意义下反映了软件的可靠性？

【解析】 突然停止正常工作的原因可能有：

用户突然启用一个以前从未用过的新功能（用户的可预测性）；

或某个软件维护人员犯了个错误，引入了一个新错误。

MTBF 是反映“用户可预测性”与“软件中存在错误数”的一个复杂函数。

【问题 3】 信息部门的吴总工程师向李工程师建议了另一类测试方案作为“错误随机植入”测试方法的补充。即由甲和乙两组测试人员同时相互独立地测试同一份程序的两个拷贝，测试了两周后，甲组发现的错误总数为 330 个，乙组发现的错误总数为 320 个，其中两个组发现的相同错误数目为 300 个。请大体上估算一下在测试前此程序原有多少个错误？并在 100 字以内简要说明使用这类估算方法的必要前提。

【解析】 测试前程序原有错误可以认为是 BNO = 320 × 330/300 = 11 × 32 = 352 个。估算的前提：

① 头几周发现的错误在全部错误中有代表性；

② 两组发现的不同错误数所占比例相对很少。

3.5 培训与用户支持

由于数据库应用系统本身的复杂性，为了使用户能够正确、方便地使用数据库系统，定期的用户培训和售后服务是必不可少的。本节主要讲述用户培训和产品运行过程中的技术支持。如图 3-10 所示是本节的知识框图。

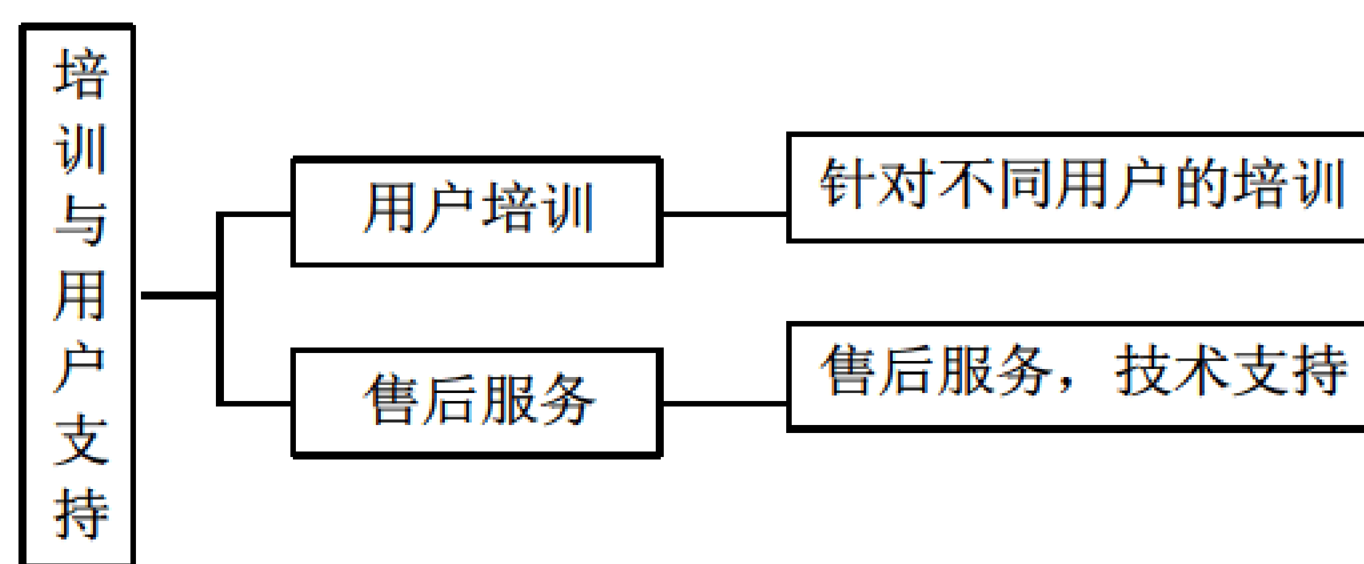


图 3-10　培训与用户支持知识框图

（1）用户培训

为了使用户能够正确、方便地使用数据库系统，除了良好的用户界面设计和简单明了的用户手册之外，定期的用户培训是必不可少的。

根据使用系统的内容和权限，可以将用户分为各级管理者、业务雇员和外部用户 3 大类。

① 各级管理者。这里的管理者通常是指从业务经理上至总经理等管理层，他们主要关心系统的统计数据。因此对各级管理者的培训内容主要集中在统计数据的查询和使用上。为了整个系统的安全，培养用户的安全意识也是很必要的。

② 业务雇员。业务雇员是与系统打交道最多的人。一般来说，业务雇员的变动性较大，对业务不熟悉，而且计算机操作水平有限，他们必须经过必要的培训和考核后才能正确而有效地使用数据库。业务雇员是培训的主要对象。培训的内容应该包括熟悉业务流程及规范、掌握应用程序的操作、培养用户的安全意识等。

③ 外部用户。对于开放的数据库系统，会有企业之外的人员进行访问，甚至是修改数据库的内容，如电子商务、网上银行中的客户。对于外部用户，可以提供用户手册以供用户查阅，并且在系统设计和实施时，尽量把用户界面做得简单友好。

（2）售后服务

由于数据库应用系统本身的复杂性，无论是 DBMS 供应商还是应用系统开发商，都不可能保证自己的产品不会有任何问题，最终用户也不可能完全有能力解决在使用过程中系统出现的问题，所以良好的售后服务不仅关系到企业的信誉，而且是系统正确有效运行的保证。

售后服务通常包括如下方面。

① 成立专门的客户服务机构，解决用户的技术问题。

② 用户培训。

③ 优惠的系统升级。

练习题

1．什么是软件测试，软件测试的目的是什么？

2．集成测试，又叫组装测试，分为两种：一次性组装和增殖式组装。根据你自己的项目经验，谈谈什么是一次性组装和增殖式组装。

3．确认测试过程要做的工作包括：有效性测试、软件配置复审、验收测试和安装测试。在验收测试中常用的有α 测试和β 测试。根据你自己的项目经验，谈谈什么是α 测试和β 测试。

4．软件测试的方法大体可以分为两种：黑盒测试和白盒测试。根据你自己的项目经验，谈谈什么是黑盒测试和白盒测试。

5．列举软件测试的一般原则。

6．软件测试并不等于程序测试。软件测试应贯穿于软件定义与开发的整个期间。请列举软件测试的对象。

7．阅读以下有关 SQL Server 2000 系统配置的叙述，回答问题。

在 SQL Server 2000 中，数据库必须至少包含一个数据文件和一个事务日志文件。数据和事务日志信息从不混合在同一文件中，并且每个文件只能由一个数据库使用。

SQL Server 使用各数据库的事务日志来恢复事务。事务日志是数据库中已发生的所有修改和执行每次修改的事务的一连串记录。事务日志记录了每个事务的开始，并记录了在每个事务期间，对数据的更改及撤销所做更改（以后如有必要）所需的足够信息。对于一些大的操作（如 CREATE INDEX），事务日志则记录该操作发生的事实。随着数据库中对记录的操作的增多，日志会不断地增长。

【问题】 根据你自己的项目经验，谈谈对创建事务日志文件的一般原则。

8．要测试某数据库产品，它能处理 1～65 535 个记录，试划分其测试等价类。

9．阅读以下有关单元测试的叙述，回答问题。

单元测试又称模块测试，是针对软件设计的最小单位——程序模块，进行正确性检验的测试工作。其目的在于发现各模块内部可能存在的各种差错。

在单元测试时，测试者需要依据详细设计说明书和源程序清单，了解该模块的 I/O 条件和模块的逻辑结构，主要采用白盒测试的测试用例，辅之以黑盒测试的测试用例，使之对任何合理的输入和不合理的输入，都能鉴别和响应。单元测试的内容如图 3-11 所示。

在图 3-11 中，测试出错处理应该考虑哪几方面的内容？

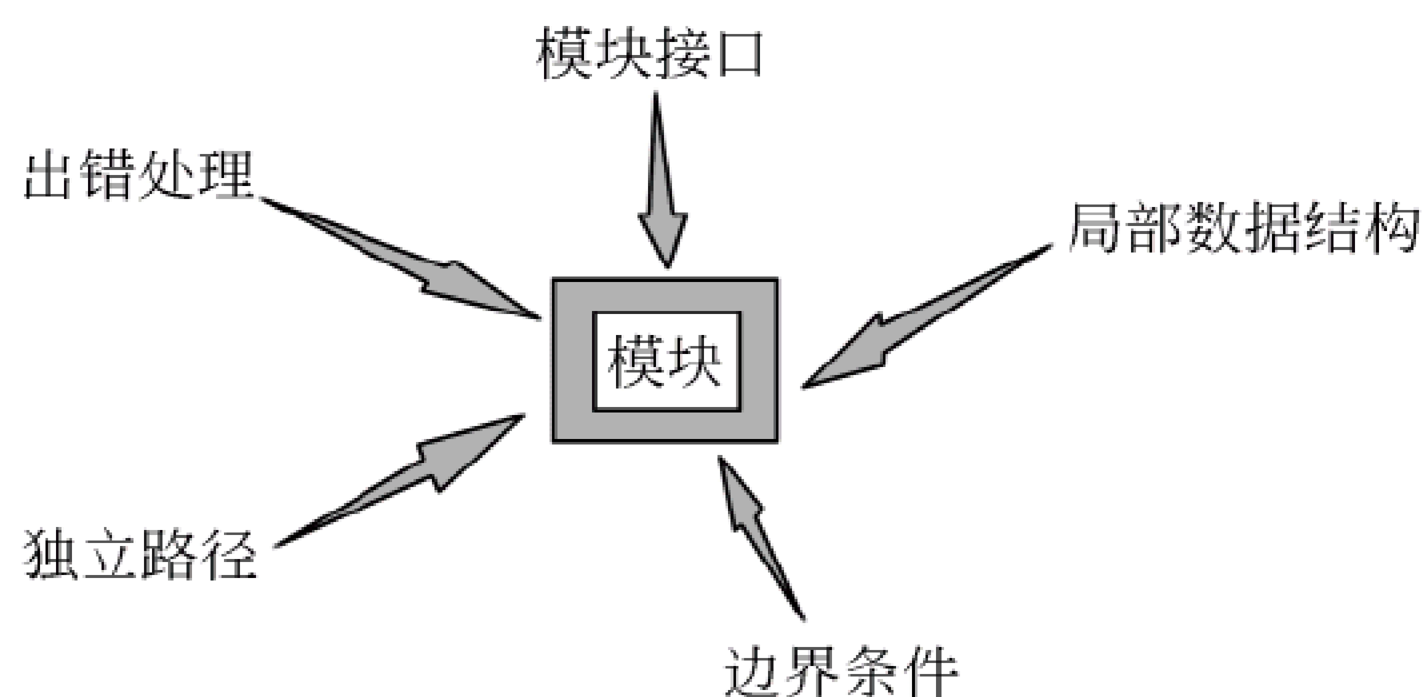


图 3-11　单元测试的内容

练习题答案

1．**【解析】** 在软件程序编码完成之后、软件投入使用之前要进行软件测试，软件测试是为了发现错误而执行程序的过程。也就是根据软件开发各阶段的规格说明和程序的内部结构而精心设计测试用例，用这些用例去运行程序以发现程序错误的过程。软件测试的

目的在于发现错误。

2.【解析】 一次性组装方式即把经单元测试后的模块一次性地组装成系统进行测试。增殖式组装方式即在模块组装的过程中，边组装边测试，每增加一个或几个模块就测试一次，最后组装成完整的系统，它又分为：自顶向下的增殖、自底向上的增殖、混合增殖等几种方式。

3.【解析】 进行α测试时，开发者坐在用户旁边，随时记录用户发现的问题。β测试则是开发者不在测试现场，它是在开发者无法控制的环境下进行的测试，通常由软件开发者向用户散发β版软件，然后收集用户的意见。

4.【解析】 黑盒测试即把测试的对象看成一个黑盒子，不考虑程序内部的逻辑结构和内部特性，主要在软件的接口处进行测试，主要测试软件的功能。黑盒测试的方法包括：等价类划分法、边界值分析、错误推测法、因果图、功能图等。

白盒测试把测试对象看成是一个打开的盒子，程序内部的逻辑结构和其他信息对测试人员都是公开的。白盒测试的方法有：逻辑覆盖（语句覆盖、判定覆盖、判定-条件覆盖、条件组合覆盖、路径覆盖）、基本路径测试等。

5.【解析】 软件测试一般有如下原则：

① 应当把“尽早地和不断地进行软件测试”作为软件开发者的座右铭；

② 测试用例应由测试输入数据和对应的预期输出结果两部分组成；

③ 程序员应避免检查自己的程序；

④ 在设计测试用例时，应当包括合理的输入条件和不合理的输入条件；

⑤ 充分注意测试中的群集现象。经验表明，测试后程序中残存的错误数目与该程序中已发现的错误数目成正比；

⑥ 严格执行测试计划，排除测试的随意性；

⑦ 应当对每一个测试结果做全面检查；

⑧ 妥善保存测试计划、测试用例、出错统计和最终分析报告，为程序维护提供方便。

6.【解析】 需求分析、概要设计、详细设计以及程序编码等各阶段所得到的文档，包括需求规格说明、概要设计规格说明、详细设计规格说明以及源程序，都应成为软件测试的对象。

7.【解析】 创建事务日志文件的一般原则有如下几点。

① 将事务日志创建在物理上单独的磁盘或 RAID（独立磁盘冗余阵列）设备上。将事务日志文件按序列写入，因此使用单独的专用磁盘可使磁头保持在下一个写入操作的位置。

② 将事务日志文件的初始大小设置为合理大小，以防止当需要更多的事务日志空间时文件自动扩展。当事务日志扩展时，将创建一个新的虚拟日志文件，并且写入事务日志的操作在事务日志扩展时会等待。如果事务日志扩展得过于频繁，性能可能会受到不良影响。

③ 将文件增长的百分比设置为合理的大小，以防止文件按太小的值增长。如果文件增长幅度与写入事务日志的日志记录数相比太小，则事务日志可能需要始终扩展，因而将

妨害性能。

④ 手工收缩事务日志文件，而不是允许 SQL Server 2000 自动收缩文件。在繁忙的系统上，收缩事务日志可能会因数据页的移动和锁定而妨害性能。

8.**【解析】** 划分 3 个测试等价类如下。

① 等价类 1：少于 1 个记录。

② 等价类 2：1～65 535 个记录。

③ 等价类 3：多于 65 535 个记录。

等价类 1 和 3 为无效等价类。

9.**【解析】** 出错处理的测试应考虑以下几点：

① 出错的描述是否难以理解；

② 出错的描述是否能够对错误定位；

③ 显示的错误与实际的错误是否相符；

④ 对错误条件的处理正确与否；

⑤ 在对错误进行处理之前，错误条件是否已经引起系统的干预等。

第 4 章　数据库系统的运行和管理

数据库系统开发完毕后，在投入使用之前，要制定合理的运行计划、系统管理计划和维护计划。数据库投入使用后，由于数据库被频繁地访问，数据不断地更新，因而各种各样影响数据库的性能和稳定的情况就会出现。本章主要讲述如何保证数据库系统安全、稳定地运行，如何解决运行中出现的各种问题，如何调整使得数据库系统发挥更大的效能。本章根据大纲的要求，全面介绍了数据库系统的运行和维护、数据库管理、数据库性能调整以及用户培训、售后服务等主要知识点。在对典型例题详细的分析之后，还给出了适量的练习题，以帮助读者加深对这些内容的理解和掌握。

本章的数据库系统的运行和维护、数据库管理和性能优化中的内容是考试中容易遇到的，考生应注意。例如在数据库系统的运行和维护过程中，新旧系统的转换时数据迁移应该注意的问题，数据库重构时的注意点，安全视图的管理，运行统计数据的分析；数据库管理环节中 OLAP 和 OLTP 的异同和选型，数据物理存放时的数据分段、分区管理，联机事务系统中事务的并发控制和死锁监测等；性能调整中利用数据库管理系统特定功能对 SQL 语句进行优化、表设计中的规范化和反规范化、索引的改进等等。如图 4-1 所示是本章的知识框图。

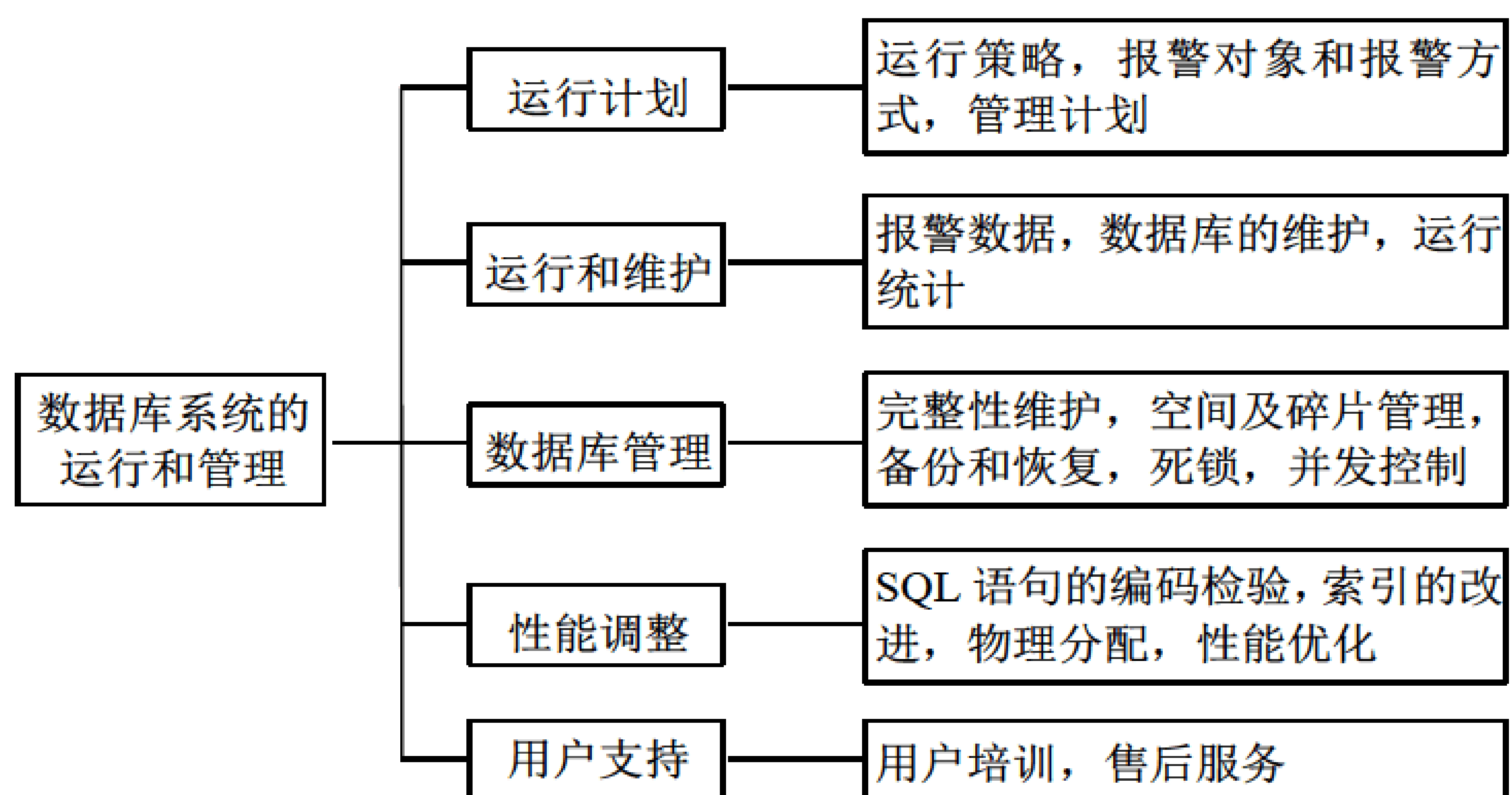


图 4-1　数据库系统的运行和管理知识框图

4.1　数据库系统的运行计划

数据库系统开发完毕后，在投入使用之前，要制定合理的运行计划，制定出数据库系统的管理计划，如执行、故障和恢复、安全性、完整性、用户培训和维护等。如图 4-2 所

示是本节的知识框图。

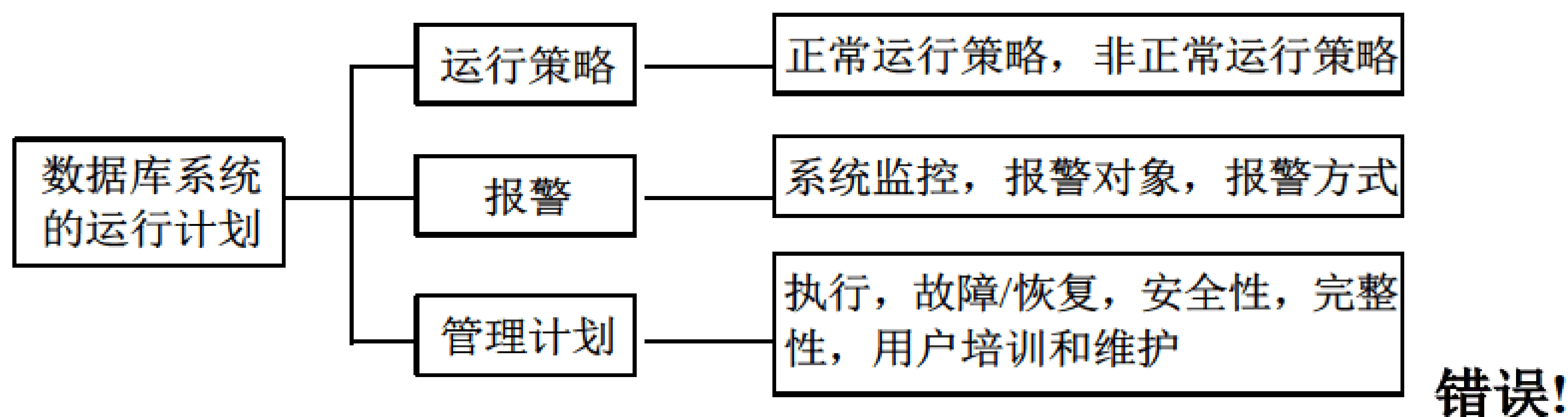


图 4-2　数据库系统的运行计划知识框图

1．知识点提炼

（1）运行策略的确定

数据库设计的最后阶段是数据库的运行与管理，包括维护数据库的安全性与完整性、监测并改善数据库性能，在必要时需要进行数据库的重新组织和构造。为了保证数据库系统安全、稳定地运行，需要综合考虑系统运行时可能遇到的各种问题，并制定出详尽的运行计划和应对措施，即制定运行策略。

运行策略的制定要从两个方面考虑：正常运行策略和非正常运行策略。

一般情况下，任何一个系统都有相对固定的用户群和访问量，系统的负载相对稳定，这种状态称为正常运行状态。正常运行策略就是指数据库系统在正常运行状态下的执行策略。正常运行策略需要考虑以下几个方面。

① 系统运行对物理环境的要求。系统的物理环境是保障系统稳定运行的基本条件。物理环境包括系统运行场所的温度、湿度、通风条件、灰尘指标、电力供应等。

② 系统运行对管理人员的要求。作为企业运行中的数据库，需要有专门的数据库管理人员来管理和维护。在实际运行中，需要根据数据库的规模和管理的难易确定管理人员的数量。对于小型的数据库系统，可以由一名或者几名数据库管理员来管理。对于比较大型的数据库系统，需要成立数据库运行管理机构，专门负责数据库系统的运行。

③ 数据库的安全性策略。数据库在运行期间会有用户的访问和操作，要实施安全性策略来保护数据库以防止不合法的使用所造成的数据泄露、更改或破坏。安全性策略可以从物理环境、网络环境、操作系统、数据库系统和人员管理等方面考虑，包括网络安全、用户的权限管理、设备的安全、数据的安全等多个方面。

④ 数据库备份和恢复策略。在数据库的运行过程中，难免会出现计算机系统的软、硬件故障，这些故障会影响数据库中数据的正确性和完整性，甚至破坏整个数据库系统，使数据库中的全部或部分数据丢失。因此需要对数据库进行备份。系统出现故障之后，利用数据库的备份文件，及时使数据库恢复到故障前的正确状态。另外，数据库系统运行过程中，数据会不断地变化和增长。某些系统会产生大量的数据，如果不及时将这些数据从系统中导出，系统的存储设备就会很快被占满，从而影响系统的正常运行，甚至使系统瘫

痪。因此，为了保证数据库系统的正常运行，需要根据系统的业务量，制定相应的数据备份策略，定期将数据从系统中导出。同时，备份数据库也是系统故障恢复所必需的。

系统运行不可能一直处于正常运行状态，在各种因素的影响下，系统会处于特殊的运行时期，例如系统的访问量可能会出现激增。非正常运行策略就是指特殊时期的数据库运行策略。非正常运行策略需要考虑以下两个方面。

① 突发事件的应对策略。突发事件是指在数据库正常运行时突然发生断电、设备故障等事件，甚至是火灾、水灾等不可抗拒的自然灾害。对于突发事件，必须预先制定及时的应对策略，如在突发事件发生后自动启用备用电源和备用设备，使系统能够保持正常运行。

② 高负载运行状态的应对策略。当业务量较多时，访问数据库的人次和操作会有明显的增加，甚至是激增，这时数据库系统运行在高负载状态。有些高负载状态是可以预计的，有些是事先难以预计的。因此，在设计数据库的负载时，要比正常运行时的负载有一定的冗余量。此外，针对系统的高负载运行状态，要设计必要而有效的应对策略，进行系统负载平衡。

（2）确定数据库系统的报警对象和报警方式

数据库系统中需要监控的对象有系统性能、系统故障和系统安全。在对系统进行监控的同时，可以设定出现严重问题时的系统报警，及时通知数据库管理员采取必要的措施，以保障数据库系统稳定地运行。

根据监控对象的不同，系统监控可分为性能监控、故障监控、安全监控。与监控对象相对应，系统报警对象包括性能报警、故障报警和安全报警。

① 性能监控是掌握系统运行性能的重要手段，性能监控需要监控的指标主要有资源占用率、用户量、事务响应时间、事务量、死锁等。当性能指标出现异常时，要及时发出性能报警信号，通知数据库管理员采取应对措施，及时将系统指标恢复到正常范围，保证系统正常、稳定地运行。

② 故障监控是保障数据库系统正常运行的手段。根据系统故障的类型不同，需要分别监控事务故障、系统故障和介质故障，当出现故障时，必须及时发出故障报警信号，通知数据库管理员进行干预，及时排除故障，恢复数据库。

③ 安全监控是对破坏数据库安全的事件的监控，安全监控主要包括入侵监控、用户访问监控和病毒监控等。当监测到非法访问数据库或者非法入侵时，应及时发出安全报警信号，数据库管理员要采取有效的措施，防范和制止非法使用数据库，确保数据库的安全性和完整性。

根据监控方式的不同，数据库系统的监控可分为系统监控和应用程序监控。

① 系统监控是指通过 DBMS 提供的监控功能，设定参数后，由系统自动进行监控。DBMS 软件都在不同程度上提供了监控功能，数据库管理员可以有效地利用。

② 应用程序监控是指根据具体情况编制应用程序来对系统进行监控，是对 DBMS 监

控功能的有益补充。

另外，系统日志是系统监控的主要依据。日志文件中详细记录了系统运行中的各种信息，管理员可以从日志文件中了解系统的运行状态和事件，并以此为根据发现系统运行中的问题。

2．难点分析

（1）数据库系统的管理计划（执行、故障/恢复、安全性、完整性、用户的培训和系统的维护）

在数据库系统运行过程中，需要对系统进行有效的管理，要制定详细的管理内容和计划。数据库系统的管理计划主要包括运行管理、故障/恢复管理、安全性管理、完整性管理、用户的教育与培训以及系统的维护。

运行管理的主要手段是进行性能监控，掌握系统运行性能。影响和决定系统性能的主要指标有资源占用率、用户量、事务响应时间、事务量和死锁等。为了保证系统的正常运行，必须采取正确的执行策略，使系统的主要指标处于正常范围内，保证系统运行的正确性和稳定性。

故障/恢复管理包括数据库的备份和恢复，即及时对数据库进行备份、转存系统数据、监控系统故障，并在系统发生故障之后，利用数据库的备份文件和系统日志，迅速将数据库恢复到故障前的正确状态。

安全性管理是指按照设计阶段提供的安全规范和故障恢复规范，检查系统的安全是否受到侵犯，并根据用户的实际需要授予用户相应的操作权限。在数据库运行过程中，由于应用环境发生变化，对安全性的要求也可能发生变化，数据库管理人员要根据实际情况及时调整相应的授权和密码，以保证数据库的安全性。

完整性管理是为了保证数据的正确性和相容性。随着应用环境的改变，数据库的完整性约束机制也需要做出相应的改变，管理员要对其进行调整，以满足用户的要求。另外，为了确保系统在发生故障后，能够迅速地恢复，管理员要针对不同的应用要求制定不同的转储计划，定期对数据库和日志文件进行备份，以使数据库在发生故障后能够恢复到某种一致性状态，从而保证数据库的完整性。

为了使用户能够正确、方便地使用数据库系统，除了良好的用户界面设计和简单明了的用户手册之外，定期的用户培训也是必不可少的。根据使用系统的内容和权限，可以将用户分为各级管理者、业务雇员和外部用户 3 大类。用户培训主要针对内部人员，即管理者和业务雇员。培训的内容应该包括熟悉业务流程及规范、掌握应用程序的操作、培养用户的安全意识等。

（2）数据库安全策略的确定

网络用户的安全责任：该策略可以要求用户每隔一段时间改变其口令；使用符合一定准则的口令；执行某些检查，以了解其账户是否被别人访问过等。重要的是，凡是要求用户做到的，都应明确地定义。

系统管理员的安全责任：该策略可以要求在每台主机上使用专门的安全措施、登录标题报文、监测和记录过程等，还可列出在连接网络的所有主机中不能运行的应用程序。

检测到安全问题时的对策：当检测到安全问题时应该做什么，应该通知谁，这些都是在紧急的情况下容易忽视的事情。

3．典型例题

【例题 4-1】 阅读以下有关运行策略的叙述，回答问题。

一般情况下，任何一个系统都有相对固定的用户群和访问量，系统的负载相对稳定，这种状态称为正常运行状态。正常运行策略就是指数据库系统在正常运行状态下的执行策略。

【问题】 正常运行策略需要考虑哪几个方面？

【解析】 正常运行策略一般需要考虑如下几个方面。

① 系统运行对物理环境的要求。物理环境包括系统运行场所的温度、湿度、通风条件、灰尘指标、电力供应等；

② 系统运行对管理人员的要求；

③ 数据库的安全性策略。数据库的运行离不开用户的访问和操作，安全性策略包括网络安全、用户的权限管理、设备的安全、数据的安全等方面；

④ 数据库备份和恢复策略。

【例题 4-2】 阅读以下有关数据库系统监控的叙述，回答问题 1 和问题 2。

在数据库系统运行过程中，数据库管理员需要及时了解数据库的运行状态，掌握运行状态中的各种指标，为改进系统提供依据。对系统运行状态的了解采用监控的手段。

【问题 1】 数据库系统监控的对象分别是系统性能、系统故障和系统安全，依照监控对象的不同，系统监控分为性能监控、故障监控、安全监控。性能监控、故障监控、安全监控的监控对象有哪些？

【解析】 性能监控、故障监控、安全监控的监控对象如下所述。

① 性能监控是掌握系统运行性能的手段。性能监控应当从资源占用率、事务响应时间、事务量、死锁、用户量等方面实现。

② 故障监控是保障数据库系统正常运行的手段。从数据库系统故障的类型入手，监控事务故障、系统故障和介质故障，出现需要管理员干预的故障时及时恢复。

③ 安全监控是对破坏数据库安全的事件的监控，包括入侵监控、用户访问监控、病毒监控等。

【问题 2】 对于一般的数据库管理员，系统监控的主要依据是什么？

【解析】 系统日志是监控中的主要依据。日志文件详细记录了系统运行中的各种信息，管理员可以从日志文件中了解系统的运行状态和事件，以此为据发现系统运行中的问题。

【例题 4-3】 某单位一信息项目的数据库采用了 SQL Server 7 作为后端平台，前端选择了 Borland 公司的 Delphi 及 Model Maker 作为开发工具，采用 Client/Server 模式。在开发过程中发现：SQL Server 默认的用户连接数是 15，一旦使用某数据库服务器的人多了的

时候，特别是一些用户喜欢打开多个连接的时候，经常造成超过连接数而使一些人连接不上。SQL Server 有许多默认值是可以修改的，请修改 SQL Server 用户连接数到 35，要求写出数据库的基本操作步骤。（sp_configure 语法：sp_configure [_name[,_value]]）。

【解析】 在 SQL Server 中，以 user connection 标记用户数目，所以做如下修改。

① 以管理员账号登录进 SQL Server 数据库服务器。

② 运行 sp_configure 系统存储过程：在 Isq_w 或 Enterprise Manager 中的 SQL Query Tool 中输入以下语句。

```
sp_configure "user connections", 35
go   --最大只能是 32 767
reconfigure
go
```

③ 执行输入的 SQL 语句。

④ 关闭 SQL Server 服务器并重新启动。因为 sp_configure 所带的参数分两类，动态与静态，动态参数不需要重新启动服务器，运行 sp_configure 和 reconfigure 后就改变了，而静态参数要重新启动后才能改变。上面的 user connections 就是静态参数。

【例题 4-4】 阅读以下关于 Oracle 数据库安全方面的叙述，回答问题。

Oracle 数据库作为大型数据库管理系统，近年来一直占有世界上高端数据库的最大份额，其强大而完善的数据库管理功能，以及 Oracle 公司推陈出新的不断努力，一直成为 IT 业界瞩目的焦点。某单位一信息项目的数据库采用了 Oracle 7.3 作为后端平台，前端选择了 Oracle 公司的 Developer 2000 及 Designer 2000 作为开发工具，采用了 Client/Server 模式。

【问题】 该系统的数据库管理员是大学毕业不久的小王，他对 Oracle 等大型数据库的管理还不是很有经验，请就如何保证数据库的安全列举相应的策略。

【解析】 要注意实施以下方面的安全策略。

① 修改 sys、system 的口令。

② 删除默认用户：dbsnmp、ctxsys 等。

③ 把 REMOTE_OS_AUTHENT 改成 False，防止远程机器直接登录。

④ 把 OS_DICTIONARY_ACCESSIBILITY 改成 False。

⑤ 把一些权限从 Public Role 取消掉。

⑥ 检查数据库数据文件的安全性，不要设置成 666 之类的，检查其他 DBA 用户。

⑦ 把一些不需要的服务（比如 FTP、NFS 等）关闭掉。

⑧ 限制数据库主机上面的用户数量。

⑨ 定期检查 Metalink/OTN 上面的 Security Alert。比如：http://otn.oracle.com/deploy/security/alerts.htm。

⑩ 把数据库与应用放在一个单独的子网中，否则用户密码很容易被窃听。或者采用 advance security，对用户登录加密。

限止只有某些 IP 才能访问数据库。

lsnrctl 要加密码，要不然别人很容易从外面关掉 listener。

如果可能，不要使用默认 1521 端口。

4.2　数据库系统的运行和维护

一般来说，数据库系统的运行和维护在整个系统的生命周期中占有很大的比例，系统的运行和维护时间越长，则系统的生命周期就越长，系统的使用价值也越大。可见，数据库的运行和维护工作是很重要的。如图 4-3 所示是本节的知识框图。

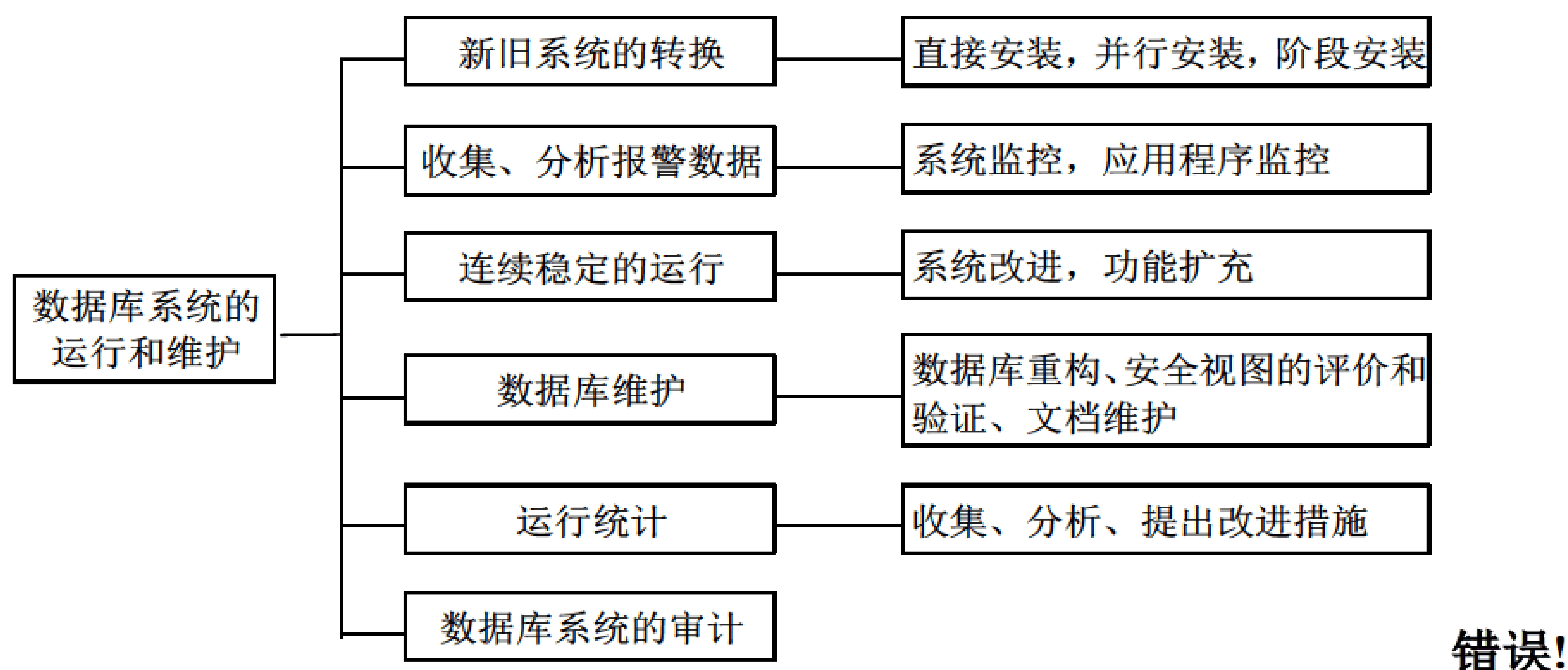


图 4-3　数据库系统的运行和维护知识框图

1．知识点提炼

（1）新旧系统的转换

新旧系统的转换方式有以下 3 种。

① 直接安装。直接启用新系统，终止旧系统并取代旧系统运行，安装新系统并使新系统快速进入运行状态。

② 并行安装。新旧系统并行运行一段时间，然后再根据新系统的运行状况决定何时终止旧系统的运行。

③ 阶段安装。经过一系列的步骤和阶段使新系统各部分逐步投入运行，分步骤、有计划地完成从旧系统向新系统的过渡。

（2）收集和分析报警数据（执行报警、故障报警、安全报警）

根据监控方式的不同，数据库系统的监控可分为系统监控和应用程序监控。

① 系统监控是指通过 DBMS 提供的监控功能，设定参数后，由系统自动进行监控。DBMS 软件都在不同程度上提供了监控功能，数据库管理员可以有效地利用。

② 应用程序监控是指根据具体情况而编制应用程序来对系统进行监控，是对 DBMS 监控功能的有益补充。

另外，系统日志是系统监控的主要依据。日志文件中详细记录了系统运行中的各种信息。

数据库系统中需要监控的对象有系统性能、系统故障和系统安全。在对系统进行监控的同时，可以设定出现严重问题时的系统报警，及时通知数据库管理员采取必要的措施，以保障数据库系统稳定地运行。

根据监控对象的不同，系统监控可分为性能监控、故障监控、安全监控。与监控对象相对应，系统报警对象包括性能报警、故障报警和安全报警。

① 性能监控是掌握系统运行性能的重要手段，性能监控需要监控的指标主要有资源占用率、用户量、事务响应时间、事务量、死锁等。当性能指标出现异常时，要及时发出性能报警信号，通知数据库管理员采取应对措施，及时将系统指标恢复到正常范围，保证系统正常、稳定地运行。

② 故障监控是保障数据库系统正常运行的手段。根据系统故障类型的不同，需要分别监控事务故障、系统故障和介质故障，当出现故障时，必须及时发出故障报警信号，通知数据库管理员进行干预，及时排除故障，恢复数据库。

③ 安全监控是对破坏数据库安全的事件进行监控，安全监控主要包括入侵监控、用户访问监控和病毒监控等。当监测到非法访问数据库或者非法入侵时，应及时发出安全报警信号，数据库管理员要采取有效的措施，防范和制止非法使用数据库，确保数据库的安全性和完整性。

（3）连续稳定地运行

数据库管理员必须采取各种措施保证数据库系统能够连续稳定地运行。目前大多数 DBMS 产品都提供了监测系统性能参数的工具，数据库管理员可以利用系统提供的这些工具，经常对数据库的存储空间状况、事务的响应时间、业务量等重要指标进行分析评价，尽量早发现系统潜在的各种问题，并有针对突发事件的应急措施。

为了使数据库系统能够连续稳定地运行，要做到以下 3 点。

① 结合用户的反应情况确定改进措施，按用户的要求对数据库现有的功能进行适当地扩充。

② 及时改正运行中发现的错误。

③ 在增加新功能时应保证原有功能和性能不受损害。

（4）数据库系统的运行统计（收集、分析、提出改进措施）

数据库的系统监控和系统运行统计是了解数据库系统运行状态的最有效手段，系统监控通常用来保障系统的稳定运行，系统的运行统计则是用来了解系统性能，作为系统性能调整的重要依据。

系统的运行统计既可以通过 DBMS 提供的工具来实现，也可以使用第三方软件。一般

来说，统计数据可以以图形、表格等多种形式直观地表示出来，并给出相应的分析结果。通过统计数据，数据库管理员可以了解系统的性能和资源占用情况，实施系统改进和资源配置，以提高系统的执行性能。

数据库的运行统计既可以是长期的，也可以是阶段性的。比如对访问量的统计是长期的，峰值时期的统计则是阶段性的。

（5）数据库系统的审计

数据库审计提供用户使用数据库资源情况的记录。通过分析审计记录，可对影响系统安全的访问和访问企图进行事后分析和追查。从分析中还可以获得有关系统安全状况的信息，为改善和加强系统安全，发现和处理可疑事件提供决策信息。根据数据库系统特点，应实现选择性审计，对审计内容及粒度应根据需要进行配置。

2．难点分析

（1）数据库维护（数据库重构、安全视图的评价和验证、文档维护）

数据库建立后，除了数据本身是动态变化以外，随着应用环境的变化，数据库本身也必须变化以满足应用要求。数据库系统运行一段时间以后，由于记录的不断增加、删除和修改会改变数据库的物理存储结构，使数据库的物理特性受到破坏，从而降低了数据库存储空间的利用率和数据的存取效率，导致数据库的性能下降。因此，需要对数据库进行重新组织，简称重组，即重新安排数据的存储位置，回收垃圾应用的存储空间，减少指针链，改进数据库的响应时间和空间利用率，提高系统性能。数据库的重组只是使数据库的物理存储结构发生变化，而数据库的逻辑结构不变。所以，根据数据库的三级模式，可以知道数据库重组提高了系统的性能，不会改变原设计的数据逻辑结构和物理结构，所以对系统功能没有影响。

数据库应用环境的变化可能导致数据库的逻辑结构发生变化，比如要增加新的实体，增加某些实体的属性，这样实体之间的联系发生了变化，致使原有的数据库设计不能满足新的要求，因此必须对原来的数据库进行重新构造，适当调整数据库的模式和内模式，比如要增加新的数据项、增加或删除索引、修改完整性约束条件等。

一般地，DBMS 都提供重新组织和构造数据库的应用程序，用来帮助数据库管理员完成数据库的重组和重构工作。

只要数据库系统在运行，就需要不断地进行修改、调整和维护。如果应用需求变化太大，数据库重新构造也无济于事，这就表明数据库应用系统的生命周期结束，需要建立新的系统，重新设计和实现数据库。

数据库重构包括表结构的修改和视图的修改。

表结构的修改包括数据列的增删和修改、约束的修改、表的分解与合并。DBMS 有一定的逻辑独立性，因此其中有些需要修改应用程序，有些则不需要，分别如下。

① 修改属性的列名或数据类型。修改表中属性的列名或数据类型必须修改使用该表的应用程序，所以应尽量少做这样的修改。

② 数据列的增加和删除。增加和删除属性只修改使用该列的应用程序。

③ 约束的修改。如果是 DBMS 支持的约束，比如主码约束、参照完整性约束和检查约束，一般来说并不需要修改应用程序，复杂的约束可以通过修改触发器程序实现。

④ 表的分解。表的分解可以通过建立并分解与表同名的视图来避免修改应用程序，但是这样修改会引起性能的下降。如果是为了提高性能而对表进行分解，则需要修改应用程序，只访问分解后的一个表。

⑤ 表的合并。表的合并通常是为了提高系统性能，可以通过建立两个与原表同名的视图来避免应用程序的修改。

视图机制一方面可以实现数据的逻辑独立性，另一方面可以实现数据的安全性，将不允许应用程序访问的数据屏蔽在视图之外。由于在数据库的重构过程中引入或修改视图，可能会影响数据的安全性，因此必需对视图进行评价和验证，确保不会因为数据库的重构而引起数据的泄密。

文档是对系统结构和实现的描述，在系统设计开发和维护过程中起着非常重要的指导作用。文档必须与系统保持高度的一致性，对系统所做的任何修改都必须以文档的形式记录下来，否则会给系统的维护带来困难或使其出现错误，甚至会危及系统的生命。所以，在对数据库重新组织和重新构造时进行的所有修改，必须在文档中体现出来。

（2）数据库系统的审计

前面介绍的各种数据库安全性措施，都可将用户操作限制在规定的安全范围内。然而实际上任何系统的安全性措施都不是绝对可靠的，所以对于某些高度敏感的保密数据，必须使用审计功能作为预防手段。

审计是 DBMS 提供的一种监视工具，它记录数据库资源和权限的使用情况。启用审计功能，可以跟踪记录有关数据的访问活动，包括哪些数据库对象受到了影响，谁在什么时候执行了这些操作。审计追踪把用户对数据库的所有操作自动记录下来，存放在一个特殊文件中，即审计日志（Audit Log）中。

审计记录的内容一般包括：

① 操作类型，如修改、查询等；

② 操作终端标识与操作者标识；

③ 操作日期和时间；

④ 操作所涉及到的相关数据，如基本表、视图、记录、属性等；

⑤ 数据的前象和后象。

利用记录的这些信息，可以重现导致数据库现有状况的一系列事件，以进一步找出非法存取数据的人、时间和内容等。从这个角度来说，审计是被动的，它只能跟踪对数据库的修改而不能防止或者阻止对数据库的非法操作，但作为一种安全性手段，可以对非法入侵起到一定的威慑作用，可以据此追究非法入侵者的法律责任。

使用审计功能会大大增加系统的开销，审计功能的开启会影响到系统的执行性能，尤

其是在一个负载比较大的系统中，会导致系统性能明显降低。另外，审计跟踪信息会被自动保存下来，占用大量的存储空间。解决这一问题的方法是根据不同级别有选择地进行审计，使其对存储和性能的负面影响降到最小，例如，可以在数据库级别、数据库对象级别和用户级别进行审计。

通常，DBMS 将审计作为一个可选特征，提供相应的操作语句可以灵活地打开或者关闭审计功能。

3．典型例题

【例题 4-5】 数据库实现阶段主要做哪些事情？

【解析】 数据库实现阶段主要做以下几个方面的工作：

① 数据的载入和应用程序的调试；

② 数据库的试运行；

③ 数据库的运行和维护。

【例题 4-6】 在 Oracle 数据库中，以下的脚本反映了数据库管理员做了什么维护工作？

```
SQL>SET serveroutput ON
BEGIN
dbms_output.enable(10000);
FOR bk_ts IN (SELECT DISTINCT t.ts#,t.name FROM v$tablespace t,v$datafile d
WHERE t.ts#=d.ts#) LOOP
dbms_output.put_line('--'||bk_ts.name);
dbms_output.put_line('ALTER TABLESPACE '||bk_ts.name||' BEGIN BACKUP;');
for bk_file in (SELECT file#,name FROM v$datafile WHERE ts#=bk_ts.ts#) LOOP
dbms_output.put_line('HOST CP '||bk_file.name||' $BACKUP_DEPT/');
END LOOP;
dbms_output.put_line('ALTER TABLESPACE '||bk_ts.name||' END BACKUP;');
END LOOP;
END;
```

【解析】 快速得到整个数据库的热备脚本。

【例题 4-7】 阅读以下有关对数据库进行重新组织的叙述，回答问题。

小刘是某信息系统的数据库管理员。在该数据库运行一段时间后，由于记录的不断增加、删除和修改，改变了数据库的物理存储结构，使数据库的物理特性受到破坏，从而降低了数据库存储空间的利用率和数据的存取效率，使数据库的性能下降。因此，信息处的王科长建议小刘对数据库进行重新组织，即重新安排数据的存储位置，回收垃圾占用的存储空间，减少指针链，改进数据库的响应时间和空间利用率，提高系统性能。

【问题】 小刘担心数据库重新组织以后，数据库后台和前台的程序也要做相应的修改。请根据自己的项目经验谈谈对此的看法。

【解析】 数据库的重组只是使数据库的物理存储结构发生变化，而数据库的逻辑结构

不变，所以根据数据库的三级模式，可以知道数据库重组对系统功能没有影响，只是为了提高系统的性能。

【例题 4-8】 某单位一信息项目的数据库采用了 DB2 作为后端平台，前端选择了 IBM 公司的 WebSphere 作为开发工具，采用 B/S 结构模式。某天，在数据库维护过程中发现：在做数据库备份操作时遇到 DB2 备份历史文件 db2rhist.asc 损坏， 错误代码为 SQL2048。试解决该问题，要求写出数据库的基本操作步骤。

【解析】 可按照如下步骤操作：

① 先在 NODE000x/SQL000x 目录下找到该文件，及其备份文件 db2rhist.bak，将 db2rhist.asc 文件移动到其他目录中。

② 重新运行备份命令。此命令仍会报告同样的错误，但同时 DB2 系统会自动用 db2rhist.bak 重新生成 db2rhist.asc 文件。

③ 再次运行备份命令，它将能够正确运行。

【例题 4-9】 DBMS 提供的审计功能有什么作用？

【解析】 使用如下 SQL 语句打开对表 S 的审计功能，对表 S 的每次成功查询、增加、删除、修改操作都做审计追踪，例如：

```
AUDIT SELECT, INSERT, DELETE, UPDATE,
ON S WHENEVER SUCCESSFUL
```

要关闭对表 S 的审计功能可以使用如下语句：

```
NO AUDIT ALL ON S
```

【例题 4-10】 阅读以下关于数据表维护方面的叙述，回答问题 1 和问题 2。

小李是某单位一信息项目的数据库管理员。在该项目的数据库管理过程中，小李发现表 source_table 在设计的时候没有建立组合字段 field_name1 和 field_name2 的唯一约束，现在需要增加这一约束时发现由于某种原因不能直接增加约束，于是做了如下操作。

① 生成组合字段重复的临时表 source_dup_simple，编写并执行如下的 SQL 语句。

```
CREATE TABLE source_dup_simple
NOLOGGING
PCTFREE 1 PCTUSED 99
AS SELECT field_name1,field_name2,COUNT(0) AS num FROM source_table
GROUP BY field_name1,field_name2 HAVING COUNT(0)>1;
```

② 编写并执行如下的 SQL 语句。

```
CREATE TABLE source_dup
NOLOGGING
PCTFREE 1 PCTUSED 99
```

```
AS SELECT t1.* FROM source_table t1,source_dup_simple t2
WHERE t1.field_name1=t2.field_name1 AND t1.field_name2=t2.field_name2;
```

③ 编写并执行如下的 SQL 语句。

```
DELETE FROM source_dup a WHERE rowid > (
SELECT MIN(rowid) FROM source_dup b WHERE
a.field_name1 = b.field_name1 AND a.field_name2=b.field_name2);
COMMIT;
```

④ 编写并执行如下的 SQL 语句。

```
DELETE FROM source_table
WHERE field_name1||field_name2 IN (SELECT field_name1||field_name2 FROM
source_dup_simple);
COMMIT;
```

⑤ 编写并执行如下的 SQL 语句。

```
INSERT INTO source_table SELECT * FROM source_dup;
COMMIT;
```

【问题 1】 不能直接增加约束的原因是什么？

【解析】 表里已经有了很多重复记录了。

【问题 2】 进行以上操作的目的是什么？

【解析】 删除表里已有的重复记录，以便在该表上加入基于组合字段 field_name1 和 field_name2 的唯一约束。

【例题 4-11】 在 Oracle 中，审计分为用户级审计和系统级审计，试叙述它们的作用。

【解析】 用户级审计和系统级审计的作用分别为：用户级审计是任何 Oracle 用户可设置的审计，主要是用户针对自己创建的数据库表或视图进行审计，记录所有用户对这些表或视图的一切成功和不成功的访问以及各种类型的 SQL 操作；系统级审计只能由 DBA 设置，用以监测成功或失败的登录要求、监测 GRANT 和 REVOKE 操作以及其他数据库级权限下的操作。

【例题 4-12】 阅读以下有关 Oracle 审计功能的叙述，回答问题 1 和问题 2。

Oracle 的审计功能很灵活，是否使用审计，对哪些表进行审计，对哪些操作进行审计等都可以由用户选择。在 Oracle 中，审计设置以及审计内容均存放在数据字典中。其中审计设置记录在数据字典表 SYS.TABLES 中，审计内容记录在数据字典表 SYSAUDIT_TRAIL 中。为此，Oracle 提供了 AUDIT 语句设置审计功能，NOAUDIT 语句取消审计功能。设置审计时，可以详细指定对哪些 SQL 操作进行审计。

【问题 1】 审计语句 AUDIT ALTER, UPDATE ON SC 的含义是什么？

【解析】 对修改 SC 表结构或数据的操作进行审计。

【问题 2】 审计语句 NOAUDIT ALL ON SC 的含义是什么？

【解析】 取消对 SC 表的一切审计。

4.3 数据库管理

数据库管理的内容比较宽泛，本节主要讲述了数据字典和数据仓库的管理、数据完整性维护和管理、数据库物理结构的管理、数据库空间及碎片管理、备份和恢复、死锁管理、并发控制、数据安全性管理，最后概括了数据库管理员的职责。如图 4-4 所示是本节的知识框图。

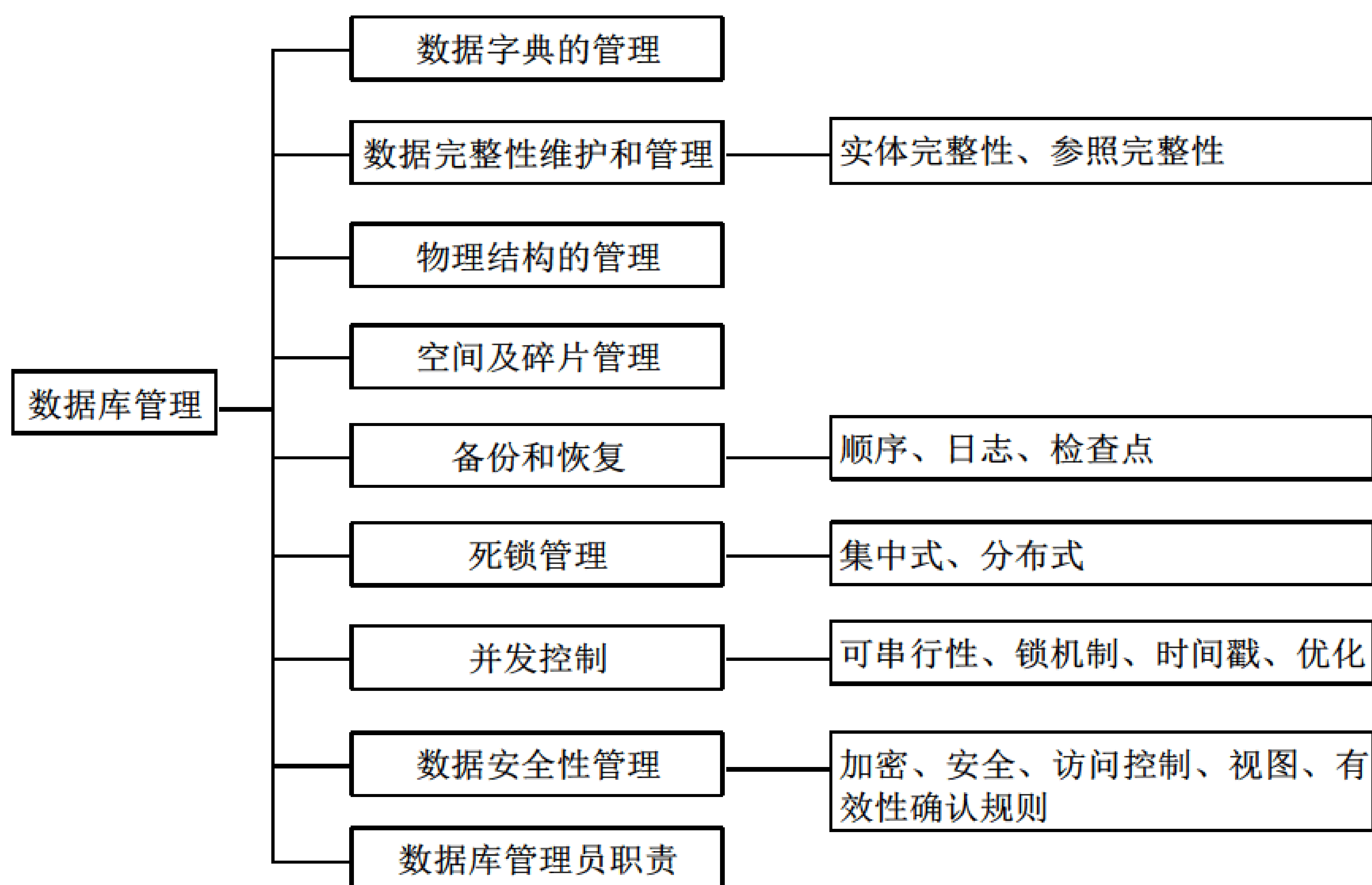


图 4-4 数据库管理知识框图

1．知识点提炼

（1）数据字典的管理

数据字典是存储在数据库中的所有对象信息的知识库，存有用户信息、用户的权限信息、所有数据对象、表的约束条件、统计分析数据库的视图等信息。数据字典通常包括数据项、数据结构、数据流、数据存储和处理过程 5 个部分。通常把数据字典中存储的数据称为元数据，是用来描述数据的，如字段的类型、长度等信息。数据本身将存放在物理数据库中，由数据库管理系统管理。数据字典有助于这些数据的进一步管理和控制，为设计人员和数据库管理员在数据库设计、实现和运行阶段控制有关数据提供依据。系统对数据库的访问，是通过数据字典中元数据所提供的信息来访问具体数据的，同时也可以通过元

数据了解数据库对象的信息。

当使用 DDL 语言定义数据库对象或使用某些 DML 语言进行表扩展等操作时，系统会自动修改数据字典中的元数据。数据字典是只读的，不同的 DBMS 提供相应的数据字典访问命令，可能通过这些命令访问数据字典的内容。在系统运行阶段，对数据库对象结构信息的修改会自动地反映到数据字典中，这些修改可能会影响应用程序对数据库的正确访问，所以需要通过数据字典的信息对应用程序做相应修改。

（2）数据完整性维护和管理（实体完整性、参照完整性）

数据库的完整性是指数据语言上的一致性和数据的相容性。DBMS 提供了完整性约束机制，当对数据进行修改时由系统对修改数据进行完整性检查，防止错误数据进入数据库。完整性约束条件作用的对象可以是表、行和列。列级约束主要是指对列的类型、取值范围、精度、非空值、值不可重复等的约束条件。行级约束是记录字段值之间联系的约束条件。表级约束是表的主码约束、表与表间的参照完整性约束、表中记录间的联系约束。

数据的完整性不仅可以通过 DBMS 系统提供的完整性约束机制来实现，也可以通过应用程序来实现，以保证运行过程中数据的正确性。在系统运行过程中，对数据完整性的维护和管理主要采取如下两种方式。

① 通过 DBMS 系统提供的完整性约束机制来实现。对于 DBMS 管理的约束，通过修改数据库的定义，如增加或删除实体完整性约束、参照完整性约束、检查约束来实现。

② 通过应用程序来实现。应用程序可以实现复杂的完整性约束，通常是用触发器程序来实现对数据库的完整性管理。

（3）数据库物理结构的管理

在数据库系统运行过程中，随着数据的不断变更，数据的物理存储会发生变化，进而影响到系统的响应效率。为了提高系统性能，保证数据不推迟访问，可以采取以下策略进行存储管理。

① 将索引文件和数据文件分开存储，把事务日志文件存储在高速设备上；

② 适时修改数据文件和索引文件的页面大小；

③ 增加必要的索引项；

④ 定期对数据进行排序；

⑤ 定期对数据进行转储。

除了对数据库的存储进行有效地管理之外，还可以通过增加计算机内存、使用高速存储设备等外部方式来提高系统的访问效率。

（4）数据库空间及碎片管理

数据库空间的使用情况是影响数据库性能的一个重要因素。在数据库运行过程中，数据库管理员要经常观察数据库表空间的使用率和剩余大小，了解当前空间的使用状况，并进行必要的调整。

许多大型 DBMS 需要每天 24 小时，每年 365 天不间断地运行。由于数据库应用系统

每天对数据库进行大量的插入、更新、删除等操作，在数据库的物理存储介质上产生了大量存储碎片，从而影响了存储的效率以及数据库应用系统运行的速度。因此需要进行碎片管理。数据在存储空间上排列得越紧密有序，数据库服务的访问速度就越快，消除碎片有助于提高系统的性能及更有效地利用数据存储空间，相对于增加硬件而言，空间及碎片管理是一种低成本的保持数据库性能的途径。

（5）备份和恢复（顺序、日志、检查点）

在数据库系统运行过程中，可能会发生故障而破坏数据的安全性和完整性，甚至破坏整个数据库系统，使数据库中全部或部分数据丢失。因此需要对数据库进行备份。系统出现故障之后，利用数据库的备份文件，及时使数据库恢复到故障前的正确状态。

数据库系统可能发生的故障大致可以分为事务故障、系统故障和介质故障 3 类。随着存储设备稳定性的不断提高，硬件故障发生的概率越来越小，介质故障几乎可以忽略，故障主要集中在由应用程序引起的事务故障和系统故障上。目前大多数的 DBMS 都提供了备份和恢复机制，在数据库系统运行过程中，系统管理员需要做的工作主要是做好数据库备份和日志文件管理。

对数据库进行备份时要注意以下几点：

① 根据数据变更情况，设定合理的备份周期和备份时间；

② 根据业务量和数据的多少，在必要时对数据进行转存储；

③ 事务日志文件保存在最稳定的存储设备上；

④ 定期在事务日志文件中加入检查点（Checkpoint）。

（6）死锁管理（集中式、分布式）

一般来说，多用户数据库管理系统都提供了并发控制机制，以实现事务的并发调度并进行死锁管理。实际运行中的数据库系统，死锁的产生往往是因为事务程序的错误。管理员通过系统监控工具或系统日志，找出频繁产生死锁的事务，分析死锁的原因，修改事务程序来减少死锁，提高系统的并发性。

（7）数据安全性管理（加密、安全、访问控制、视图、有效性确认规则）

一般的大型数据库加密包括数据库本身的加密和通信数据加密两种。数据库加密技术是将数据库中的原始数据（明文）转换成人们所不能识别的数据（密文），达到防止信息泄漏的目的。加密方案应保证对入侵者事实上不可破解，而合法用户存取数据时因加解密而付出的空间和时间开销较小。譬如对于有次序关系而且多用于查询条件的属性采用同态加密算法，以保持数据之间的大小关系。

数据库加密通常需要大量的密钥，数据及其相应密钥的生命周期较长，因而密钥管理是实现密文数据库的关键技术。密钥管理包括密钥的产生、存储、分发、删除、获取和应用。现代密钥管理采取多级安全密钥管理体制。

数据库系统的安全性管理应该从以下几个方面考虑：

① 建立网络级安全，主要是防火墙的设置；

② 建立操作系统级安全，对用户进行登录管理；

③ 建立 DBMS 级安全，对访问数据库的用户进行密码验证；

④ 用户的授权管理；

⑤ 建立视图和存储过程加强安全性；

⑥ 使用审计功能，为追纠非法入侵者法律责任提供证据，发现安全漏洞。

（8）数据库管理员职责

数据库管理员（DBA）需要根据企业的数据库制定相应的策略和政策，并为之提供必要的技术支持。作为一名合格的数据库管理员，不仅需要对 DBMS 内部的技术有很好的了解，而且需要具有深厚的计算机软、硬件方面的知识。

数据库管理员的职责如下。

① 定义概念模式。确定对企业有用的实体和实体的相关信息，创建概念模式，即逻辑设计。

② 定义内模式。创建存储结构定义，定义概念模式/内模式映像，即物理设计。

③ 与用户联络。确定用户的数据需求，定义外模式，外模式/概念模式映像。

④ 设计应用程序。

⑤ 定义安全性和完整性约束。

⑥ 定义转储和重载机制。数据库中存放了重要数据，数据库易受破坏。转储就是对数据库备份，用于在发生故障时重载，将数据库恢复到先前的正确状态。

⑦ 监控系统性能并响应不断变化的需求。在数据库系统运行过程中，数据库管理员需要及时了解数据库的运行状态，掌握运行状态中的各种指标，为改进系统提供依据。对系统运行状态的了解采用监控的手段。

⑧ 为用户提供技术培训，帮助用户决定和解决技术问题等。

2．难点分析

（1）备份和恢复（顺序、日志、检查点）

使用日志文件进行数据库恢复时，恢复子系统必须搜索日志，确定哪些事务需要 Redo，哪些事务需要 Undo。一般来说，恢复数据库时需要检查所有日志记录，这样做有两个缺点：一是搜索整个日志将耗费大量的时间；二是很多需要 Redo 处理的事务实际上已经将它们的更新操作结果写到数据库中了，然而恢复子系统又重新执行了这些操作，浪费了大量时间。为了解决这些问题，具有检查点的恢复技术应运而生，即在原先的日志文件中增加一类新的记录——检查点记录（Checkpoint），增加一个重新开始文件，重新开始文件用来记录各个检查点记录在日志文件中的地址，并让恢复子系统在登录日志文件期间动态地维护日志。检查点记录了数据库的正确状态点，使用检查点方法可以改善恢复效率。在数据库恢复过程中，就可以反向扫描日志文件，找到第一个检查点，执行 Undo 和 Redo 操作，减少恢复的时间开销。当事务 T 在一个检查点之前提交，T 对数据库所做的修改一定都已写入数据库，写入时间是在这个检查点建立之前或在这个检查点建立之时。这样，在进行恢

复处理时，没有必要对事务 T 执行 Redo 操作。

检查点记录的内容包括：

① 建立检查点时刻所有正在执行的事务清单；

② 建立检查点时刻所有正在执行的事务最近一个日志记录的地址。

动态维护日志文件的方法是周期性地执行建立检查点、保存数据库状态的操作。具体步骤如下。

① 将当前日志缓冲中的所有日志记录写入磁盘的日志文件中。

② 在日志文件中写入一个检查点记录。

③ 将当前数据缓冲中的所有数据记录写入磁盘的数据库中。

④ 把检查点记录在日志文件中的地址写入一个重新开始文件。

恢复子系统可以定期或不定期地建立检查点，保存数据库状态。检查点既可以按照预定的一个时间间隔建立，如每隔一小时建立一个检查点；也可以按照某种规则建立，如日志文件写满一半时便建立一个检查点。

（2）数据安全性管理——加密

上面介绍的几种数据库安全措施，都是防止从数据库系统窃取保密数据，不能防止通过不正常渠道非法访问数据库。例如，窃取存储数据的磁盘，或在通信线路上窃取数据等。为了防止这些窃密活动，较好的措施是对数据加密。数据加密是防止数据库中数据在存储和传输中失密的有效手段。

数据加密的基本思想是根据一定的算法将原始数据（术语为明文，Plain text）加密成为不可直接识别的格式（术语为密文，Clipher text），数据以密文的形式存储和传输。

数据加密方法有两种，如下。

① 替换方法。该方法使用密钥（Encryption Key）将明文中的每一个字符转换为密文中的一个字符。

② 转换方法。该方法将明文中的字符按不同的顺序重新排列。

通常是将这两种加密方法结合起来使用，可以达到相当高的安全程度。数据加密后，对于不知道解密算法的人，即使利用系统安全措施的漏洞非法访问数据，也只能看到一些无法辨认的二进制代码，无法得到数据的真实信息。合法的用户检索数据时，首先提供密码钥匙，由系统进行译码后，即可得到可识别的数据信息。但是，用密码存储数据，在存入时需要加密，在查询时需要解密，这个过程会占用大量系统资源，降低了数据库的性能。

目前许多数据库产品提供了数据加密例行程序，用户可根据要求自行进行加密处理，还有一些未提供加密程序的产品也提供了相应的接口，用户可以用其他厂商的加密程序对数据进行加密。数据加密功能通常允许用户自由选择，对那些保密要求特别高的数据，才值得采用加密措施。

（3）并发控制（可串行性、锁机制、时间戳、优化）

事务的执行方式分为两种：串行方式和并行方式。

串行执行即每个时刻只有一个事务在运行，其他事务必须等到这个事务结束以后才能运行。并发方式是指多个事务同时对同一数据进行操作。

并行方式又可以分为交叉并发方式（interleaved concurrency）和同时并发方式（simultaneous concurrency）。交叉并发方式是指事务的并发操作轮流交叉运行，常见于单处理机系统中。同时并发方式是指在多处理机系统中，每个处理机可以运行一个事务，多个处理机可以同时运行多个事务，实现多个事务真正意义上的并行运行。一般来说，并发执行方式是指交叉并发方式。

对多用户并发存取同一数据的操作不加控制可能会存取和存储不正确的数据，破坏数据库的一致性，因此 DBMS 系统必须提供并发控制机制。并发控制机制是衡量一个 DBMS 性能的重要标志之一。

多个事务的并发执行是正确的，当且仅当其结果与按某一次序串行地执行它们时的结果相同时，我们称这种调度策略为可串行化（Serializable）的调度。可串行性（Serializability）是并发事务正确性的准则。按这个准则规定，一个给定的并发调度，当且仅当它是可串行化的，才认为是正确调度。可串行化调度方法有加锁控制（Locking）、乐观假设（Optimistic）和时间戳（timestamping）3 种。目前 DBMS 普遍采用封锁方法实现并发操作调度的可串行性，从而保证并行调度的正确性。

在某事务对某个数据操作之前，先对其加锁，其他事务不能更新加锁的数据，直到释放加锁为止，该技术称为封锁技术。使用封锁技术，给数据项加的锁主要有两种，如下。

① 共享锁（S）：如果事务 T 获得了数据项 Q 上的共享锁（记为 S），则 T 可读 Q 但不能写 Q。即允许其他事务对同一数据进行检索，但不得对同一数据进行修改操作，其他事务对加了共享锁的数据可读不可写；

② 排他锁（X）：如果事务 T 获得了数据项 Q 上的排他锁（记为 X），则 T 既可读 Q 又可写 Q。即如果某事务对数据加上排他锁，该事务可读写数据，其他事务不能再对数据加任何类型的锁，也就不能读写该数据，只有等待开锁。

每个事务要根据自己将对数据项 Q 进行的操作申请适当的锁，该请求是发送给并发控制管理器的。只有在并发控制管理器授予所需要的锁之后，事务才能继续其操作。

在实际的数据库管理系统中，并发控制管理机制中的锁不止排他锁和共享锁，还有其他形式的锁。具体内容可以参考各商业 DBMS 的各种文档或手册。

时间戳协议是一种不同于封锁技术的并发控制协议。系统在启动事务时，为事务分配一个时标。时标既表示了事务启动的时间，同时也表明了事务的优先级。通常，时标小的表示先启动，具有高优先级。当一个事务 A 必须等待另一个事务 B 所持有的锁的时候，存在死锁的可能性。处理方法有两种：一是 wait-die（等待—死亡），如果 A 的优先级比 B 的高，那么 A 等待，B 不被处理；如果 A 的优先级比 B 低，A 被立即撤销。另一种是 wound-wait（伤害—等待），如果 A 比 B 优先级高，那么 B 立即撤销；否则 A 等待。为了使被撤销的事务的优先级不断被提高，被撤销的事务在重新启动的时候维持原始的时标不变。相比之

下，wait-die 最终能够保证每一事务都被完成，而 wound-wait 中回滚的事务会比较少。

3．典型例题

【例题 4-13】 试述关系模式上的 3 种完整性约束。

【解析】 关系模式上的 3 种完整性约束分别叙述如下。

① 实体完整性：关系模型中以主码作为唯一性标识，主码主属性不能取“无意义”或“不知道”的值，即空值。如果主值取空值，说明存在某个不可标识的实体，与现实世界中的实体是可区分的相矛盾。

② 参照完整性：关系模型中实体和实体间的联系都使用关系来描述。这就存在着关系和关系间的引用。

③ 用户定义的完整性：关系模型中针对某一具体的约束条件，它反映某一具体应用所涉及的数据所要满足的语义要求。

【例题 4-14】 数据库安全（Database Security）是指保护数据库以防止不合法的使用所造成的数据泄露、更改或破坏。在数据库安全领域，什么是保密性、完整性、可用性和可控性？

【解析】 保密性、完整性、可用性和可控性的含义如下。

保密性：信息不泄露给非授权的用户、实体或过程，或供其利用的特性。

完整性：数据未经授权不能进行改变的特性，即信息在存储或传输过程中保持不被修改、不被破坏和丢失的特性。

可用性：可被授权实体访问并按需求使用的特性，即当需要时应能存取所需的信息。网络环境下拒绝服务、破坏网络和有关系统的正常运行等都属于对可用性的攻击。

可控性：对信息的传播及内容具有控制能力。

【例题 4-15】 有一 SQL Server 2000 的 Server，配置的验证方式是混合模式，现新建立一个 ACCOUNT 用户，加入到系统的 administrator 组，这时启动查询分析器，但无论在 Server 端还是 Client 端都无法登录到 SQL Server。请问最有可能的原因是什么？

【解析】 最有可能的原因是 BUILTIN\Administrators 被删除了。

【例题 4-16】 某单位一信息项目的数据库采用了 SQL Server 2000 作为后端平台，由于某种特殊的需求，技术员小王需要将 tempdb 数据库备份和恢复。请问 tempdb 数据库可以备份和恢复吗？试说明可行性。

【解析】 因为 tempdb 在 SQL 停止时会自己删除，所以用备份数据文件的方法也不能备份。

【例题 4-17】 某单位一信息项目的数据库采用了 DB2 作为后端平台，版本为 7.x，操作系统平台为 UNIX，数据库建立时日志方式默认是循环日志模式（Circular Log），如何实施联机备份？请以 sample 数据库为例说明。

【解析】 数据库建立时日志方式默认是循环日志模式（Circular Log），这时是无法做联机备份的。所以，希望实施联机备份，首先要将日志方式改为归档日志模式（Archival Log）。

以 sample 数据库为例，可以在控制中心改变 sample 数据库的配置参数 LOGRETAIN 为 Recovery，或在命令行下用 DB2 UPDATE DB cfg FOR sample USING LOGRETAIN ON。改变此参数后，再次连接数据库会显示数据库处于备份暂挂（BACKUP PENDING）状态。这时，需要做一次对数据库的脱机备份。在控制中心选择对数据库进行脱机备份或在命令行下用 DB2 BACKUP DB sample 实施。此后数据库就可以进行联机备份了。

可以选择在控制中心中对数据库进行联机备份，或在命令行下用 db2 backup db sample online 实施。另外，对利用联机备份得到的 IMAGE 文件进行恢复时，还需要相关的日志文件。

【例题 4-18】 阅读以下关于数据库安全等级方面的叙述，回答问题 1 和问题 2。

TCSEC 又称桔皮书，即《DoD 可信计算机系统评估标准》（Trusted Computer System Evaluation Criteria），1985 年由美国国防部（DoD）正式颁布。TCSEC/TDI，即《可信计算机系统评估标准在数据库管理系统的解释（Trusted Database Management System Interpretation of TCSEC）》，简称 TCSEC/TDI，1991 年由美国 NCSC（国家计算机安全中心）颁布。TDI 又称紫皮书，是 TCSEC 在数据库管理系统方面的扩展。TDI 中定义了数据库管理系统的设计与实现中需满足和用以进行安全性级别评估的标准。

TCSEC/TDI 将系统划分为 4 组（division）7 个等级，依次是 D；C（C1，C2）；B（B1，B2，B3）；A（A1），按系统可靠或可信程度逐渐增高，如表 4-1 所示。

表 4-1　安全级别定义

安全级别	定　义
A1	验证设计（Verified Design）
B3	安全域（Security Domain）
B2	结构化保护（Structurol Protection）
B1	标记安全保护（Labeled Security Protection）
C2	受控的存取保护（Controlled Access Protection）
C1	自主安全保护（Discretionary Security Protection）
D	最小保护（Minimal Protection）

【问题 1】 Oracle 公司的 Trusted Oracle 7、Sybase 公司的 Secure SQL Sever version 11.0.6、Informix 公司的 Incorporated INFORMIX-Online/Secure 5.0 等产品都属于 B1 级数据库产品。B1 级数据库产品有什么特点？

【解析】 B1 级标记安全保护。对系统的数据加以标记，并对标记的主体和客体实施强制存取控制（MAC）以及审计等安全机制。B1 级能够较好地满足大型企业或一般政府部门对于数据的安全需求，这一级别的产品才认为是真正意义上的安全产品。满足此级别的产品前一般多冠以“安全”（Secure）或“可信的”（Trusted）字样，作为区别于普通产品的安全产品出售。

【问题 2】 现在市面上有 B2 级以上的数据库产品吗？

【解析】 目前没有 B2 级以上的数据库产品。

【例题 4-19】 有一 SQL Server 2000 的 Server，配置的验证方式是混合模式，现新建立一个域用户 account，这时启动查询分析器，无论在 Server 端和 Client 端都无法登录到 SQL Server，请问最有可能的原因是什么？

【解析】 正常对于混合模式验证的 SQL Server，Windows 的域用户是有 Login 权限的，如果新建立一个全局域用户，加入到 Windows 的指定的组（可以访问 SQL Server）中，但无法登录，是因为该用户必须注销再次登录时才会有 Login 的权限。

【例题 4-20】 为什么 Access 2000 数据库会损坏？如何修复？

【解析】 Access 数据库有可能因偶然原因而损坏，如电源电压不稳、没有正确关闭 Windows、病毒、网络服务中断等。任何影响计算机系统稳定工作的非正常因素都有可能会破坏数据库文件。如果 Access 因某种原因非正常退出，也可能导致数据库损坏。

因为数据库文件可能会损坏，Access 作为程序的一部分提供了一个修复工具。修复数据库的方法如下。

① 关闭所有打开的数据库。

② 选择“工具”/“数据库实用工具”/“压缩和修复数据库”。

③ 在“压缩数据库来源”对话框中，选择要修复的数据库。

④ 单击“修复”按钮。

在一个数据库修复以后，可能会丢失一些数据。因此，防止数据丢失的最好办法还是经常性地备份数据库文件。

【例题 4-21】 某单位一信息项目的数据库采用了 DB2 作为后端平台，如何实施已删除表的恢复（Dropped Table Recovery）？

【解析】 按照以下几个步骤进行。

① 首先数据库要能前滚恢复（数据库配置参数 logretain 或 userexit 打开）。

② 对要实施 Drop Table Recovery 的表空间(限 regular tablespace)，执行：alter tablespace 表空间名 dropped table recovery on 。

③ 用 list history dropped table all for 数据库名得到删除表的 tableid（例如 0000000000006d0000020003）和表结构的生成语句（DDL），记录 tableid 和该语句以便恢复。然后用 drop 命令删除的表中数据可以在前滚恢复时导出。

④ 恢复数据库后，如果想恢复已删除的表，在前滚时加 recover dropped table tableid to 目标目录。在该目录下被删除的表中的数据将被导出，例如导出至…/NODE0000/data 文件。利用上面提到的表结构生成语句生成被删除了的表，然后用 import 命令将数据导入表中。

【例题 4-22】 阅读以下关于数据库口令安全方面的叙述，回答问题。

在数据库安全中，口令保护是手段之一。但是由于口令所包含的信息位较少，作为一种保护机制是很有限的。一个短的和明显的口令很容易被识破。口令安全性依赖于特定系

统的实现。常用的口令攻击方法有：穷举法尝试（尝试所有可能的口令）、经验法尝试（最常用和可能的口令）、搜索口令表（破译口令文件、口令数据区）、询问用户（窃取口令、窥探口令输入）、编程截取口令（信息查找、侦察程序）。

为了防止口令攻击，设置安全的口令是必要的。安全的口令是那些很难猜测的口令。难猜测的原因是口令中不仅有字符，还有数字、标点符号、控制字符和空格，另外，还要容易记忆、至少有 7 至 8 个字符长及容易输入。

不安全的口令往往是：任何名字，包括人名、软件名、计算机名甚至幻想中事物的名字，电话号码或者某种执照的号码，社会保障号，任何人的生日，其他很容易得到的关于自己的信息，一些常用的词，任何形式的计算机中的用户名，在英语字典或者外语字典中的词，地点名称或者一些名词，键盘上的一些词，任何形式的上述词再加上一些数字。

【问题】 作为数据库系统管理员，在口令保护方面应该做些什么？

【解析】 作为系统管理员，应该定期检查系统是否存在无口令的用户，其次应定期运行口令破译程序以检查系统中是否存在弱口令，这些措施可以显著地减少系统面临的通过口令入侵的威胁。另外，系统管理员应保护好自己的口令，并要求用户定期更换自己的口令。

【例题 4-23】 阅读以下关于 Sybase 安全性方面的叙述，回答问题 1 和问题 2。

Sybase 中的安全性是依靠分层解决的，它的安全措施也是一级一级层层设置的，真正做到了层层设防。

【问题 1】 分别列举每一层的功能。

【解析】 每一层的功能如下：

第一层是注册用户许可，保护对服务器的基本存取；

第二层是存取控制，对不同用户设定不同的权限，使数据库得到最大限度的保护；

第三层是增加限制数据存取的视图和存储过程，在数据库与用户之间建立一道屏障。

【问题 2】 Sybase 基于上述层次结构的安全体系，提出几点实施安全的原则，请简述之。

【解析】 实施安全的原则有：

① 自主访问控制（Discretionary Access Control，DAC），用来决定用户是否有权访问数据库对象；

② 验证，就是保证只有授权的合法用户才能注册和访问；

③ 授权，对不同的用户访问数据库授予不同的权限；

④ 审计，监视系统发生的一切事件。

【例题 4-24】 阅读以下有关 Oracle 数据字典的叙述，回答问题。

Oracle 数据字典中，对象名称多数以 USER、ALL、DBA 为前缀。USER 视图中通常记录执行查询的账户所拥有的对象信息；ALL 视图中记录包括 USER 记录和授权至 PUBLIC 或用户的对象信息；DBA 视图包含所有数据库对象，而不管其所有者。

使用举例：

SELECT * FROM dba_data_files 查询表空间的信息（当前用户必须拥有 DBA 角色）。

SELECT owner,object_name,object_type FROM all_objects 查询某一用户下的所有表、过程、函数等信息。

【问题】 说明以下视图描述的数据字典的意义。

ALL_CATALOG

ALL_COL_COMMENTS

【解析】 ALL_CATALOG　用户能够访问所有的表、视图等对象。

ALL_COL_COMMENTS　用户能够访问所有表、视图等的列的说明。

4.4　性能调整

在数据库系统运行过程中，如何维护和提高系统的性能，是数据库管理员的主要工作之一。系统的性能一方面取决于 DBMS 的性能及其参数设定，而在指定的 DBMS 环境下，与具体的应用系统也有很大的关系，可通过调整来提高性能。本节主要讲述如何从逻辑上（如 SQL 语句的编码、索引的改进）和物理上（如物理分配的改进、设备的增强）两个方面来优化数据库性能。如图 4-5 所示是本节的知识框图。

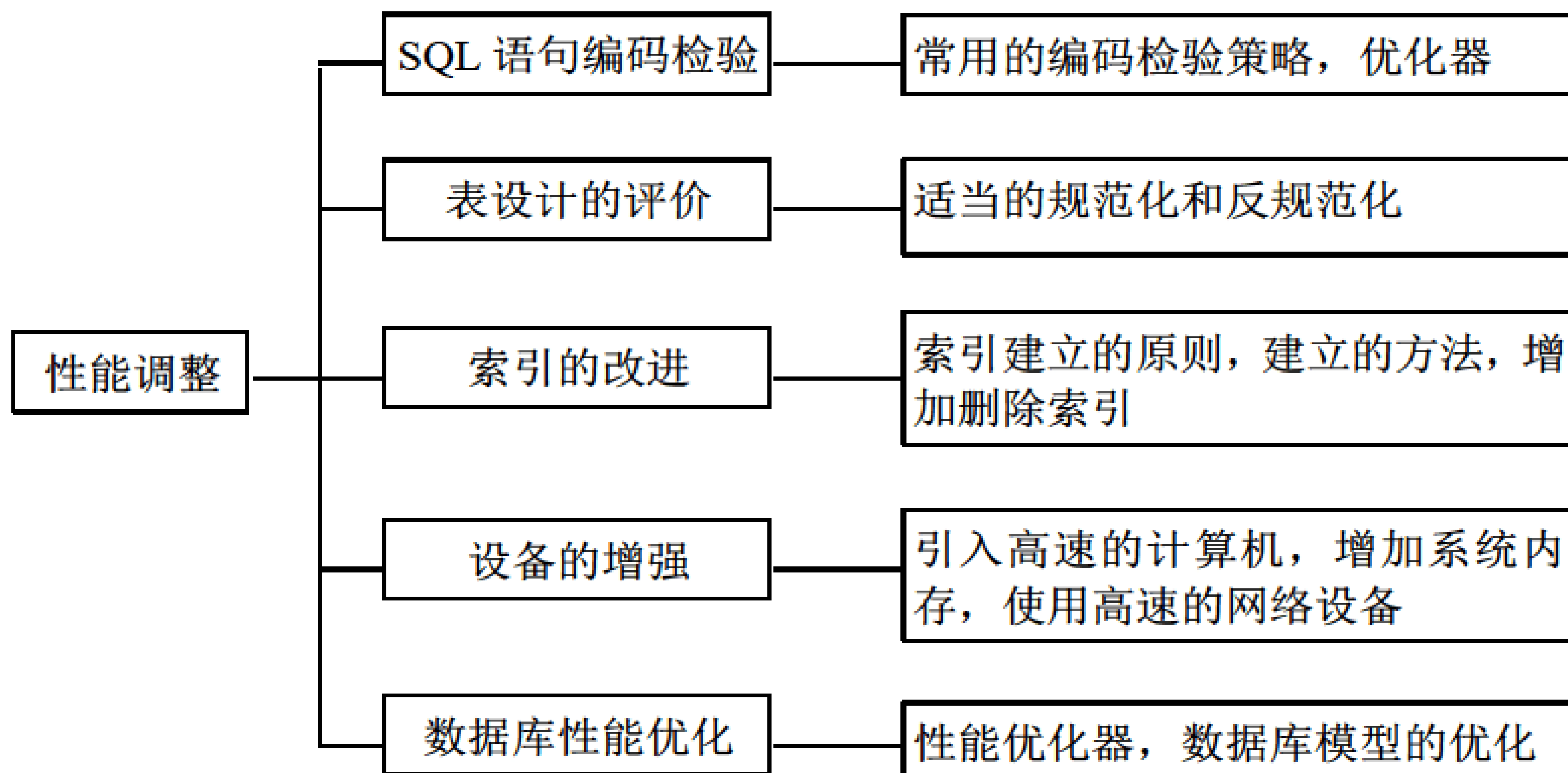


图 4-5　性能调整知识框图

1．知识点提炼

（1）SQL 语句的编码检验

通过 DBMS 提供的监控和统计功能，找出频繁执行的 SQL 语句（通常是查询语句），对其进行优化，常用的编码检验策略如下：

① 尽可能地减少多表查询或建立物化视图；

② 以不相关子查询替代相关子查询；

③ 只检索需要的列；

④ 用带 IN 的条件子句等价替换 OR 子句；

⑤ 经常提交 COMMIT，以尽早释放锁。

在应用系统开发初期，由于开发数据库数据比较少，对于查询 SQL 语句，复杂视图的编写等体会不出 SQL 语句各种写法的性能优劣，但是如果将应用系统提交实际应用后，随着数据库中数据的增加，系统的响应速度就成为目前系统需要解决的最主要的问题之一。

系统优化中一个很重要的方面就是 SQL 语句的优化。对于海量数据，劣质 SQL 语句和优质 SQL 语句之间的速度差别可以达到上百倍，可见对于一个系统不是简单地能实现其功能就可以了，而是要写出高质量的 SQL 语句，提高系统的可用性。

在多数情况下，数据库管理系统使用索引来更快地遍历表。一些大型的数据库都提供了优化器，优化器主要根据定义的索引来提高性能。但是，如果在 SQL 语句的 WHERE 子句中写的 SQL 代码不合理，就会造成优化器忽略索引而使用全表扫描。在编写 SQL 语句时应清楚优化器根据何种原则来处理索引，这有助于写出高性能的 SQL 语句。

（2）表设计的评价

在设计阶段，为了减少数据冗余和消除操作异常，提出了关系模式的设计应当符合 3NF 或 BCNF，在数据库系统运行过程中，需要根据实际情况对表进行调整。表的调整要遵循以下的原则：

① 如果频繁地对两个相关的表进行连接操作，则可以考虑将这两个表合并；

② 如果频繁地访问表中的某一部分字段，则可以考虑将表进行分解，将频繁访问的那部分字段单独作为一个表；

③ 对于更新很少的表，引入物化视图。

（3）索引的改进

改进系统中的索引可以提高系统性能。针对具体的情况，可以适当地改进索引，改进索引要遵循以下原则：

① 如果查询是瓶颈，则在关系上建立适当的索引，通常在作为查询条件的属性上建立索引，可以提高查询效率；

② 如果更新是瓶颈，每次更新都会重建表上的索引，引起效率的降低，则考虑删除某些索引；

③ 选择适当的索引类型；

④ 将有利于大多数据查询和更新的索引设为聚簇索引；

⑤ 在数据库中，用户访问的最小单位是属性。如果对某些非主属性的检索很频繁，可以考虑建立这些属性的索引文件。索引文件对存储记录重新进行内部链接，从逻辑上改变了记录的存储位置，从而改变了访问数据的入口点。关系中数据越多，索引的优越性也就越明显。建立多个索引文件可以缩短存取时间，但是增加了索引文件所占用的存储空间以及维护的开销。因此，应该根据实际需要综合考虑。

（4）设备的增强

在数据库系统运行过程中，经过各种调整和改进之后，如果系统仍然不满足性能要求，则应考虑适当地增强系统设备。设备的增强需要资金投入，应该综合考虑设备的性价比和投入产出比，还需要征得决策者的同意。

通常增强系统设备的方法有：

① 引入高速的计算机；

② 增加系统内存；

③ 使用高速的网络设备；

④ 使用高速的存储设备。

（5）数据库性能优化

对于关系型数据库系统而言，为了达到所需的性能，必须对系统性能进行优化，而关系表达式的高度语义层次使得优化能够得以进行。对于非关系型数据库系统，使用什么样的底层操作及操作次序是由用户而不是由系统决定的，所有这种系统的用户必须是编程高手，普通用户则无法从数据库系统受益。

通常，可以用优化器自动优化系统，自动优化的优势在于用户不必担心如何最好地表达他们的查询要求。

与用户手动优化系统相比，优化器具有以下优势。

① 一个好的优化器具有一些用户不具备的信息，比如每个域中值的个数、每个基本表中当前元组的数目、每个基本表中每个属性的唯一值数目以及这些唯一值在每个属性中出现的次数等。这些信息在系统目录中，可用于选择最高效的执行方案。

② 优化器可以考虑数以百计的策略，而用户一般只能考虑三、四种。

③ 优化器是包含最优秀程序员的技巧和服务的程序。

性能优化的目的是为计算给定的关系表达式选择一个高效的执行策略。优化不能保证所选择的实现策略在任何方面都是最优的。事实上，通常的优化策略只是在原来查询上的一个改进，某些条件下，可以认为是最优的。

当数据库统计信息改变时，性能优化可能需要改变执行策略。在关系型系统中，再优化很简单；而非关系型系统则需要重新编写源程序来进行系统的再优化。

数据模型的优化也是提高数据库系统性能的重要手段之一。数据库逻辑设计的结果不是唯一的，为了进一步提高数据库应用系统的性能，通常以规范化理论为指导，适当地修改、调整数据模型的结构，这就是数据模型的优化。

数据模型的优化方法如下。

① 确定数据依赖。对于各个关系模式之间的数据依赖进行极小化处理，消除冗余的联系。按照数据依赖的理论对关系模式逐一进行分析，考查是否存在部分函数依赖、传递函数依赖、多值依赖等，确定各关系模式分别属于第几范式。

② 对关系模式进行必要的分解。根据需求分析阶段得到的各种应用对数据处理的要求，分析在应用环境下这些关系模式是否合适，确定是否要对它们进行合并或分解。

2．难点分析

下面就某些 SQL 语句的 WHERE 子句编写中需要注意的问题做详细介绍。在这些 WHERE 子句中，即使某些列存在索引，但是由于编写了劣质的 SQL 语句，系统在运行该 SQL 语句时也不能使用该索引，而同样进行全表扫描，这就造成了响应速度的极大降低。

① IS NULL 与 IS NOT NULL

不能用 NULL 作索引，任何包含 NULL 值的列将都不会被包含在索引中。即使索引有多列这样的情况下，只要这些列中有一列含有 NULL，该列就会从索引中排除。也就是说，如果某列存在空值，即使对该列建索引也不会提高性能。任何在 WHERE 子句中使用 IS NULL 或 IS NOT NULL 的语句优化器是不允许使用索引的。

② 连接列

对于有连接的列，即使最后的连接值为一个静态值，优化器也是不会使用索引的。

③ 带通配符（%）的 LIKE 语句

如果通配符(%)在搜寻词首出现，数据库管理系统不使用索引。例如：SELECT * FROM student WHERE name LIKE '%c'中，数据库管理系统不会使用基于 name 列建立的索引。在很多情况下可能无法避免这种情况，一定要注意，通配符如此使用会降低查询速度。当通配符出现在字符串其他位置时，优化器就能利用索引。例如在下面的查询中索引得到了使用：SELECT * FROM student WHERE name LIKE 'c%'。

④ ORDER BY 语句

ORDER BY 语句决定了数据库管理系统如何将返回的查询结果排序。ORDER BY 语句对要排序的列没有什么特别的限制，也可以将函数加入列中（像连接或者附加等）。任何在 ORDER BY 语句的非索引项或者有计算表达式都将降低查询速度。处理该问题的办法是：仔细检查 ORDER BY 语句以找出非索引项或者表达式，它们会降低性能。解决这个问题的办法就是重写 ORDER BY 语句以使用索引，也可以为所使用的列建立另外一个索引，同时应尽量避免在 ORDER BY 子句中使用表达式。

⑤ NOT 或 <>

如果在某列上使用了 NOT 或<>等逻辑运算符，数据库管理系统将不使用基于该列的索引。例如：SELECT * FROM student WHERE age <> 20 或 SELECT * FROM student WHERE not (age = 20)等。再看下面这个例子：SELECT * FROM student WHERE age < 20 or age > 20。虽然这两种查询的结果一样，但是第二种查询方案会比第一种查询方案更快些。第二种查询中数据库管理系统对 age 列使用索引，而第一种查询则不能使用索引。

⑥ IN 和 EXISTS

有时候要将一列和一系列值进行比较，最简单的办法就是在 WHERE 子句中使用子查询。在 WHERE 子句中可以使用两种格式的子查询。

第一种格式是使用 IN 操作符：…WHERE column IN(SELECT * FROM…WHERE…)；第二种格式是使用 EXISTS 操作符：…WHERE EXISTS (SELECT 'X' FROM…WHERE…)。

相信绝大多数人会使用第一种格式，因为它比较容易编写，而实际上第二种格式要远比第一种格式的效率高。在很多的数据库管理系统中，几乎可以将所有的 IN 操作符子查询改写为使用 EXISTS 的子查询。

第二种格式中，子查询以 SELECT 'X'开始。运用 EXISTS 子句时不管子查询从表中选择什么数据，它只查看 WHERE 子句。这样优化器就不必遍历整个表而仅根据索引就可完成工作（这里假定在 WHERE 语句中使用的列存在索引）。相对于 IN 子句来说，EXISTS 使用相连子查询，构造起来要比 IN 子查询困难一些。

通过使用 EXISTS，数据库管理系统会首先检查主查询，然后运行子查询直到它找到第一个匹配项，从而节省了时间。数据库管理系统在执行 IN 子查询时，首先执行子查询，并将获得的结果列表存放在一个加了索引的临时表中。在执行子查询之前，系统先将主查询挂起，待子查询执行完毕，存放在临时表中以后再执行主查询。这也就是使用 EXISTS 比使用 IN 通常查询速度快的原因。同时，应尽可能使用 NOT EXISTS 来代替 NOT IN，尽管两者都使用了 NOT（不能使用索引而降低速度），但是 NOT EXISTS 要比 NOT IN 查询效率更高。

3．典型例题

【例题 4-25】 阅读以下关于序号方面的叙述，回答问题。

有很多应用系统要用到序号，如商场的销货号、货物入库的流水号等等。由于序号是连续生成的，在大型系统中，会有多个用户同时申请下一个序号，序号生成便成了应用系统的瓶颈问题。大型数据库都增加了序号对象。通过序号对象可以自动生成序号，多个用户可以并发读取，无须互相等待。

在 Oracle 数据库管理系统中，Oracle 序号生成器的序号生成语法为：CREATE 序号名 INCREMENT BY 每次增长数 START WITH 起始序号 。例如，生成序号 seq1，初始值为 1，每次增长为 1：CREATE seq1 INCREMENT BY 1 START WITH 1。

可以用下面的语句得到当前的序号：SELECT seq1.currval FROM dual。也可以用下面的语句得到下一个序号：INSERT INTO dept VALUES（seq1.nextval,…）。

在 Sybase 中，每个表可以有一个 identity 列，此列即为序号列。定义序号列的语法如下：

```
CREATE TABLE 表名
(…
序号列名 numeric(长度,0) identity,
…
)
```

序号列的数据类型必须是 numeric，小数位数必须是 0。

【问题】 请从序号的连续性方面叙述这两种序号机制可能对应用系统产生什么样的影响。

【解析】 在一个事务中，不管由于什么原因程序终止，已经申请的序号就不能再使用了。多个用户同时使用序号时，它们之间互不等待。因此，使用序号发生器会有跳号现象出现。

【例题 4-26】 阅读以下关于数据库方面的叙述，回答问题。

某单位一信息项目的数据库采用了 Oracle 7.3 作为后端平台，前端选择了 Oracle 公司的 Delphi 作为开发工具，采用的是 Client/Server 模式。在设计这个数据库系统时，技术员小牛遇到这样一个问题：数据库中有 tb_order、tb_item 表。根据业务要求，删除一个 order 时，相应的要把这个 order 下所有的 item 删除。在数据库中处理就是——先检查 tb_item 表，查出在这个要被删除的 order 下存在几个 item 记录，然后相应的删除这几条 item 记录，接着删除 tb_order 表中对应的记录。整个过程，如果用一个存储过程来实现，就可以比较容易地应用数据库本身的事务机制来保证数据库数据的一致性、安全特性等。

但是，为了提高程序的可重用性，小牛设计业务逻辑层时采用了面向对象的思想，分别设计一个 Order 类和一个 Item 类——把上述的完整的存储过程划分成 3 个存储过程：

删除 Order 即操作 tb_order 的存储过程 sp1；

删除 Item 即操作 tb_item 的存储过程 sp2；

检查在这个 order 下对应几个 item 的方法即操作 tb_item 的存储过程 sp3。

相应的在 Delphi 类的设计时如下定义：

Order 类中定义了 Order 的删除方法为 Order.cancel（Order.cancel 首先调用 Order.checkitem——调用存储过程 sp3，然后调用相应若干对象的 Item.cancel——调用存储过程 sp2，最后调用删除 Order 的存储过程 sp1）。

Item 类中有 Item 的删除方法为 Item.cancel（它调用存储过程 sp2）。

检查在这个 order 下对应几个 item 的方法（调用存储过程 sp3）在 Order 类中定义为 Order.checkitem。

【问题】 小牛担心这种处理会在操作数据库的时候出现异常：程序执行 Order.cancel 时，流程走到 c 处执行 sp1 时出错，那么如何回滚 Item.cancel 执行的动作呢?请给出解决这个问题的方案。

【解析】 解决这个问题的一个方法是：将 Order.cancel 写在一个大的事务内；而将 Item 和 Order 类的成员函数写在一个小的事务中。在提交小事务之前，判断是否处于大事务的调用过程中。如果是，则不提交事务，只是返回一个执行成功与否的结果，让外层的大事务提交；否则自行提交事务。

【例题 4-27】 阅读以下关于 DB2 方面的叙述，回答问题 1 和问题 2。

在设计一个 DB2 数据库系统（版本：DB2 V8.1）时，技术员小崔遇到这样一个问题：SAMPLE 数据库中有两张表，WORKDEPT 和 EMPLOYEE。WORKDEPT 的主键是 DEPTNO；EMPLOYEE 的主键是 EMPLOYEE，外键 DEPTNO 的父表是 WORKDEPT。小崔想修改 WORKDEPT 的主键 DEPTNO 字段的内容（将 DEPTNO 的 003 改为 005），发现

由于外键约束，不能修改。EMPLOYEE 的外键 DEPTNO 的字段内容也不能修改。

【问题 1】 技术员小陶就如何解决这个问题拿出方案：修改外键时，使用 CASCADE 关键字，例如：UPDATE CASCADE…，该方案的可行性如何？

【解析】 该方案不可行，因为 DB2 V8.1 中确实没有 UPDATE CASCADE 规则。

【问题 2】 请根据自己的项目经验，给出可行的方案。

【解析】 主要考虑两种方案：如果使用了 UPDATE RESTRICT 规则，再添加 DELETE CASCADE 规则，就可以先删除父表中的一条记录（子表对应的记录也会删除），然后向父表中插入一条新记录，再往子表中插入一条新记录；另一种方法是把 UPDATE RESTRICT 去掉，只用默认的 UPDATE NO ACTION 外键限制，这样可能更方便些。

【例题 4-28】 阅读以下有关系统性能的叙述，回答问题 1 和问题 2。

数据库中的数据是以文件的形式存储在物理存储设备上的，通常是磁盘系统。应用程序通过 DBMS 完成 I/O 操作来访问数据。I/O 操作的效率直接影响到系统的运行效率，提高系统访问效率的有效手段就是提高 I/O 操作的效率。

【问题 1】 提高数据库性能的最直观的办法是提高系统的硬件水平，请列举几种通过提高系统的硬件水平来提高系统性能的办法。

【解析】 在硬件方面，提高系统的访问效率有如下几个办法：

① 增加并行处理的 CPU 个数；

② 增加计算机内存容量；

③ 引入高速存储设备等。

【问题 2】 在数据库系统运行过程中，数据的不断变更会影响到系统的响应效率。通过一些手段进行存储管理，可有效地提高系统性能，请列举几种。

【解析】 提高系统性能的存储管理手段有如下几种：

① 索引文件和数据文件分开存储，事务日志文件存储在高速设备上；

② 适时修改数据文件和索引文件的页面大小；

③ 定期对数据进行排序；

④ 增加必要的索引项。

【例题 4-29】 阅读以下有关数据库碎片的叙述，回答问题 1 和问题 2。

Oracle 作为一种大型数据库，广泛应用于金融、邮电、电力、民航等数据吞吐量巨大、计算机网络广泛应用的重要部门。对于系统管理员来讲，如何保证网络稳定运行，如何提高数据库的性能，使其更加安全高效，就显得尤为重要。作为影响数据库性能的一大因素——数据库碎片，应当引起 DBA 的足够重视，及时发现并整理碎片仍是 DBA 的一项基本维护内容。

【问题 1】 当生成一个数据库时，它会分成称为表空间（Tablespace）的多个逻辑段（Segment），如系统（System）表空间、临时（Temporary）表空间等。一个表空间可以包含多个数据范围（Extent）和一个或多个自由范围块，即自由空间（Free Space）。 请从表

空间、段、范围、自由空间的组织关系方面解释数据库碎片产生的原因。

【解析】 当表空间中生成一个段时，将从表空间的有效自由空间中为这个段的初始范围分配空间。在这些初始范围充满数据时，段会请求增加另一个范围。这样的扩展过程会一直继续下去，直到达到最大的范围值，或者在表空间中已经没有自由空间用于下一个范围。最理想的状态就是一个段的数据可被存在单一的一个范围中。这样，所有的数据存储时靠近段内其他数据，并且寻找数据可少用一些指针。但是一个段包含多个范围的情况是大量存在的，没有任何措施可以保证这些范围是相邻存储的。当要满足一个空间要求时，数据库不再合并相邻的自由范围（除非别无选择），　而是寻找表空间中最大的自由范围来使用。这样将逐渐形成越来越多的离散的、分隔的、较小的自由空间，即碎片。

【问题 2】 随着时间推移，基于数据库的应用系统的广泛使用，产生的碎片会越来越多，将对数据库有两点主要影响。请列举并解释原因。

【解析】 产生的碎片会越来越多，将对数据库有以下两点主要影响。

① 导致系统性能减弱。当要满足一个空间要求时，数据库将首先查找当前最大的自由范围，而“最大”的自由范围逐渐变小，要找到一个足够大的自由范围已变得越来越困难，从而导致表空间中的速度障碍，使数据库的空间分配愈发远离理想状态；

② 浪费大量的表空间。尽管有一部分自由范围（如表空间的 pctincrease 为非 0）将会被 SMON（系统监控）后台进程周期性地合并，但始终有一部分自由范围无法得以自动合并，浪费了大量的表空间。

【例题 4-30】 阅读以下有关 SQL Server 数据库性能的叙述，回答问题 1 到问题 4。

在一个大型的数据库中，性能成为人们关注的焦点之一，如何让数据库高效地运行成为广大数据库管理人员和开发人员必须要考虑的问题。性能是一个应用或多个应用在相同的环境下运行时对效率的衡量。性能常用响应时间和工作效率来表示。响应时间是指完成一个任务花费的时间，可以从以下 3 方面来减少响应时间：

① 减少竞争和等待的次数，尤其是磁盘读写等待的次数；

② 利用更快的部件；

③ 减少利用资源所需的时间。

绝大多数性能的获得来自于优秀的数据库设计、精确的查询分析和适当的索引。为了取得更好的数据库性能，就需要对数据库进行优化，减少系统资源的竞争，如对数据高速缓存、过程高速缓存、系统资源和 CPU 的竞争等等。

在 SQL Server 中，有如下优化层次：

应用层——大部分性能的获得来自于对 SQL 应用中查询的优化，这必须是以好的数据库设计为基础的；

数据库层——应用共享在数据库层中的资源，这些资源包括硬盘、事务日志和数据 cache；

操作系统层——理想地，SQL Server 是一台机器的唯一主要应用，它必须和操作系统

以及其他 Sybase 软件，如 Backup Server 或 SQL Server Monitor 共享处理器、内存以及其他资源；

服务器层——在服务器层有许多共享的资源，包括数据高速缓存、过程高速缓存、锁、CPU 等；

网络层——指连接用户和 SQL Server 的网络。

【问题 1】 在大多数情况下，我们是对应用层进行优化，因为对应用性能的优化是最乐于接受的功能，其结果能被观测及检验。查询的性能是 SQL 应用的整个性能的一个关键。那么应用层上的优化都包括哪些方面的内容？

【解析】 应用层上的优化包括如下方面的内容：

① 远程处理或复制处理能够把决策支持从 OLTP 机器中分离出来；

② 优化数据模型，减少为了关联一致性对数据修改需要 JOIN 操作；

③ 添加适当的索引；

④ 利用存储过程来减少编译时间和网络的利用；

⑤ 利用最少量的锁去满足应用需要；

⑥ 减少不必要的安全审计。

【问题 2】 数据库层上的优化包括哪些方面的内容？

【解析】 数据库层上的优化包括如下方面的内容：

① 利用事务日志的阀值来自动转储事务日志防止其超出使用空间；

② 在数据段中用阀值来监视空间的使用；

③ 利用分区来加速数据的装入；

④ 对象的定位以避免硬盘的竞争；

⑤ 把重要表和索引放入 cache 中，以便快速读取。

【问题 3】 服务器层上的优化包括哪些方面的内容？

【解析】 服务器层上的优化包括如下方面的内容：

① 优化内存——一个关键的配置参数和其他方面的参数；

② 决策是客户端处理还是服务器端处理——有些处理是否能在客户端进行；

③ 配置 cache 的大小和 I/O 的大小；

④ 增加多个 CPU；

⑤ 为空闲时间排定批处理任务和生成报表；

⑥ 工作负荷发生改变，重新配置特定参数；

⑦ 决定是否能把 DSS 移到另一个 SQL 服务器中的设备层。

【问题 4】 设备层上的优化包括哪些方面的内容？

【解析】 设备层上的优化包括如下方面的内容：

① 使用多个中等大小的设备及多个控制器可能比用少量的大设备有更好的 I/O 性能；

② 分布数据库、表和索引以在不同的设备上进行 I/O 装载；

③ 增加 CPU 以适应工作负荷；
④ 配置调度程序以提高 CPU 利用率；
⑤ 遵循多处理器应用设计指导以减少竞争；
⑥ 配置多个数据 cache 操作系统层。

4.5　用户支持

在数据库应用系统的使用过程中，要注意对用户进行必要的培训，做好售后服务工作，使用户能够正确、方便地使用数据库系统。本节重要讲述了用户培训和售后服务等知识点。如图 4-6 所示是本节的知识框图。

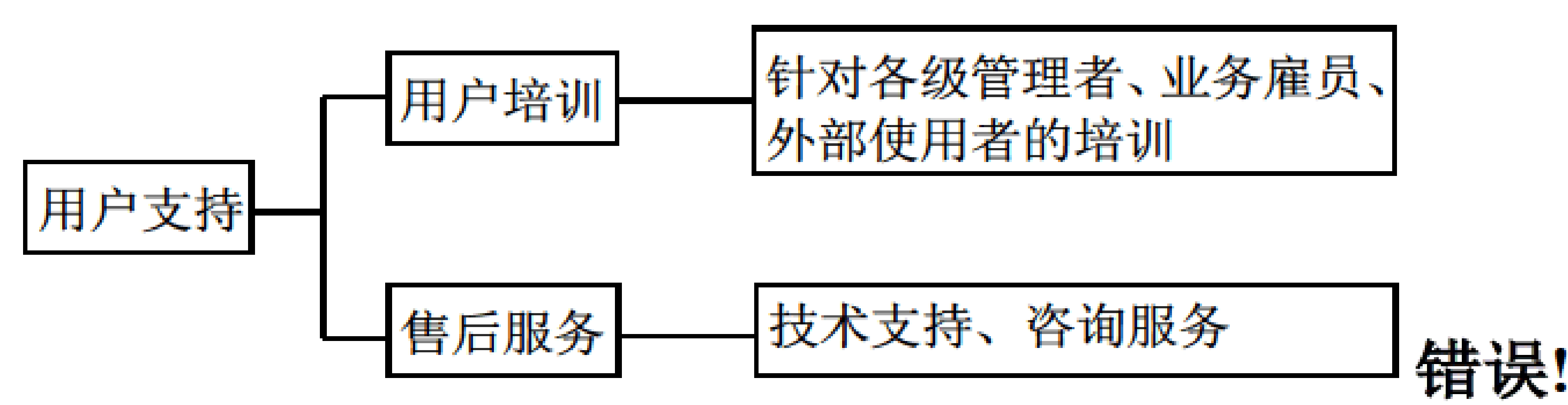


图 4-6　数据库设计知识框图

（1）用户培训

为了使用户能够正确、方便地使用数据库系统，除了良好的用户界面设计和简单明了的用户手册之外，定期的用户培训是必不可少的。

根据使用系统的内容和权限，可以将用户分为各级管理者、业务雇员和外部用户 3 大类。

① 各级管理者。这里的管理者通常是指从业务经理上至总经理等管理层，他们主要关心系统的统计数据。因此对各级管理者的培训内容主要集中在统计数据的查询和使用上。为了整个系统的安全，培养用户的安全意识也是很必要的。

② 业务雇员。业务雇员是与系统打交道最多的人。一般来说，业务雇员的变动性较大，对业务不熟悉，而且计算机操作水平有限，必须经过必要的培训和考核后才能正确而有效地使用数据库。业务雇员是培训的主要对象。培训的内容应该包括熟悉业务流程及规范、掌握应用程序的操作、培养用户的安全意识等。

③ 外部用户。对于开放的数据库系统，会有企业之外的人员进行访问，甚至是修改数据库的内容，如电子商务、网上银行中的客户。对于外部用户，可以提供用户手册以供用户查阅，并且在系统设计和实施时，就要尽量把用户界面做得简单友好。

（2）售后服务

由于数据库应用系统本身的复杂性，无论是 DBMS 供应商还是应用系统开发商，都不

可能保证自己的产品不会有任何问题，最终用户也不可能完全有能力解决在使用过程中系统出现的问题，所以良好的售后服务不仅关系到企业的信誉，而且是系统正确而有效地运行的保证。

售后服务通常包括：

① 成立专门的客户服务机构，为用户提供技术支持，解决用户的技术问题，包括热线咨询服务和上门服务；

② 用户技术培训；

③ 优惠的系统升级。

练习题

1．某单位一信息项目的数据库采用了 SQL Server 2000 作为后端平台，在企业管理器里根据向导完全备份数据库，但是在别的机器上无法恢复，最有可能是什么原因？

2．数据库系统的生存周期分为哪几个阶段？数据库结构的设计在生存期中的地位如何？

3．为什么数据库设计过程中要有一个概念设计阶段？

4．关系数据库规范化理论在逻辑设计中起什么作用？

5．回答下列问题。

【问题 1】 数据库设计过程包括几个主要过程？

【问题 2】 哪些阶段独立于数据库管理系统？

【问题 3】 哪些阶段依赖于数据库管理系统？

6．数据库安全（Database Security）是指保护数据库以防止不合法的使用所造成的数据泄露、更改或破坏。请根据自己的项目经验，列举一般的分布式数据库应该注意的典型安全问题。

7．数据库安全（Database Security）是指保护数据库以防止不合法的使用所造成的数据泄露、更改或破坏。访问控制机是数据库安全的重要组成部分，主要包括两方面：定义用户权限（授权）和合法权限检查，解释其含义。

8．数据库隔离控制技术是数据库安全的重要手段，请回答有关数据库隔离技术的问题 1 和问题 2。

【问题 1】 隔离控制技术，是数据库系统常见且比较成熟的一种存取控制方法，什么是隔离控制技术？

【问题 2】 在通常的数据库管理系统中，常用的隔离控制手段有哪两种？

9．Oracle 支持角色的概念。所谓角色就是一组系统权限的集合，目的在于简化权限管理。Oracle 除允许 DBA 定义角色外，还提供了预定义的角色，如 CONNECT、RESOURCE 和 DBA，这些角色各具有什么样的权限？

10. 试问下列关系模式最高属于第几范式，并解释其原因。

【问题 1】 R 的属性集合为{A，B，C，D}，函数依赖集合为 F={B→D，AB→C}。

【问题 2】 R 的属性集合为{A，B，C，D，E}，函数依赖集合为 F={AB→CE，E→AB，C→D}。

【问题 3】 R 的属性集合为{A，B，C，D}，函数依赖集合为 F={B→D，D→B，AB→C}。

【问题 4】 R 的属性集合为{A，B，C}，函数依赖集合为 F={A→B，B→A，A→C}。

【问题 5】 R 的属性集合为{A，B，C}，函数依赖集合为 F={A→B，B→A，C→A}。

【问题 6】 R 的属性集合为{A，B，C，D}，函数依赖集合为 F={A→C，D→B}。

【问题 7】 R 的属性集合为{A，B，C，D}，函数依赖集合为 F={A→C，CD→B}。

练习题答案

1. **【解析】**常见的错误就是创建备份设备时，没有把前面的备份设备删除掉，导致本次备份到多个设备中，而在另外的机器上恢复时只复制了所关心的备份，所以无法恢复。

2. **【解析】**数据库系统的生存周期分为以下 7 个阶段：规划、需求分析、概念设计、逻辑设计、物理设计、数据库实施、数据库运行以及维护。数据库结构的设计在生存期中处于最为重要的关键地位，是后几个阶段的基础。

3. **【解析】**数据库的设计和实施是一个极为复杂的过程，因此需要事先进行概念设计，以保证数据库有目的、有顺序。概念模型是对现实世界的抽象和概括，它真实、充分地反映了现实世界中事物和事物之间的联系，能满足用户对数据的处理要求。概念模型具有简洁和易操作的特点，因此可用于和不熟悉计算机的用户交换意见，使用户能积极参与数据库的设计工作，保证设计工作顺利进行。概念模型易于更改，当应用环境和应用要求改变时，也更容易对概念模型修改和扩充。

4. **【解析】**关系数据库规范化理论可用来改造关系模式。通过对关系模式的分解来消除其中不适合的数据依赖，以解决插入异常、删除异常、更新异常和数据冗余问题。

5. **【解析 1】**数据库设计过程包括六个主要过程：需求分析、概念结构设计、逻辑结构设计、物理设计、数据库实施、数据库运行和维护。

【解析 2】需求分析和概念结构设计独立于数据库管理系统。

【解析3】依赖于数据库管理系统的阶段有：概念结构设计、逻辑结构设计、物理设计、数据库实施、数据库运行和维护。

6. **【解析】**一般的分布式数据库应该注意的典型安全问题有如下几个方面。

信息泄露（disclosure of information）：造成将有价值的和高度机密的信息暴露给无权访问该信息的人的所有问题。

数据篡改（unauthorized access）：入侵者非法篡改数据，以获取利益或破坏数据完

整性。

拒绝服务（denial of service）：使得系统难以或不可能继续执行任务的所有问题。

7．定义用户权限（授权）和合法权限检查的含义分别如下。

定义用户权限（授权）：用户权限是指不同的用户对于不同的数据对象允许执行的操作权限。系统必须提供适当的语言定义用户权限，这些定义经过编译后存放在数据字典中，被称做安全规则或授权规则。

合法权限检查：每当用户发出存取数据库的操作请求后（请求一般应包括操作类型、操作对象和操作用户等信息），DBMS 查找数据字典，根据安全规则进行合法权限检查，若用户的操作请求超出了定义的权限，系统将拒绝执行此操作。

8．**【解析 1】**即通过某种中间机构，将用户与存取对象隔离。用户不能直接对存取对象进行操作，而是通过中间机构间接进行的。

【解析 2】常用的手段有视图和存储过程。

9．**【解析】**这些角色各具有的权限如下。

具有 CONNECT 角色的用户可以登录数据库，执行数据查询和操纵。即可以执行 ALTER TABLE、CREATE VIEW、CREATE INDEX、DROP TABLE、DROP VIEW、DROP INDEX、GRANT、REVOKE、INSERT、SELECT、UPDATE、DELETE、AUDIT、NOAUDIT 等操作。

RESOURCE 角色可以创建表，即执行 CREATE TABLE 操作。创建表的用户将拥有对该表的所有权限。

DBA 角色可以执行某些授权命令，创建表，对任何表的数据进行操纵。它涵盖了前两种角色，此外还可以执行一些管理操作，DBA 角色拥有最高级别的权限。

10．**【解析 1】**属于第一范式。因为 A、B 都是不可分的基本数据项。

【解析 2】属于第二范式。因为 F 属于第一范式并且 F 的每一个非主属性都完全函数依赖于 R 的码。

【解析 3】属于第二范式。因为 F 属于第一范式并且 F 的每一个非主属性都完全函数依赖于 R 的码。

【解析 4】属于第一范式。因为 A、B、C 通过传导成为不可分的基本数据项。

【解析 5】属于第一范式。因为 A、B、C 通过传导成为不可分的基本数据项。

【解析 6】属于 BCNF 范式。因为符合 x→y，且 y 不包含于 x，使 x 含有候选码。

【解析 7】属于第一范式。因为 C、D 是不可分的基本数据项。

第 5 章　SQL

本章提示

SQL（Structured Query Language，结构化查询语言），凭其简洁、易学和功能丰富的优点已经成为关系数据库的标准语言，利用 SQL 不仅可以查询数据库中的数据，还可以管理数据库。流行的关系数据库系统都对 SQL 做了一定的扩展，从而允许用户使用终端就可以直接操作数据库，包括管理数据库和更新数据库中的数据。我们将从定义、操作、安全机制、触发器和 SQL 的使用方式等几个方面学习 SQL。如图 5-1 所示的是本章的知识框图。

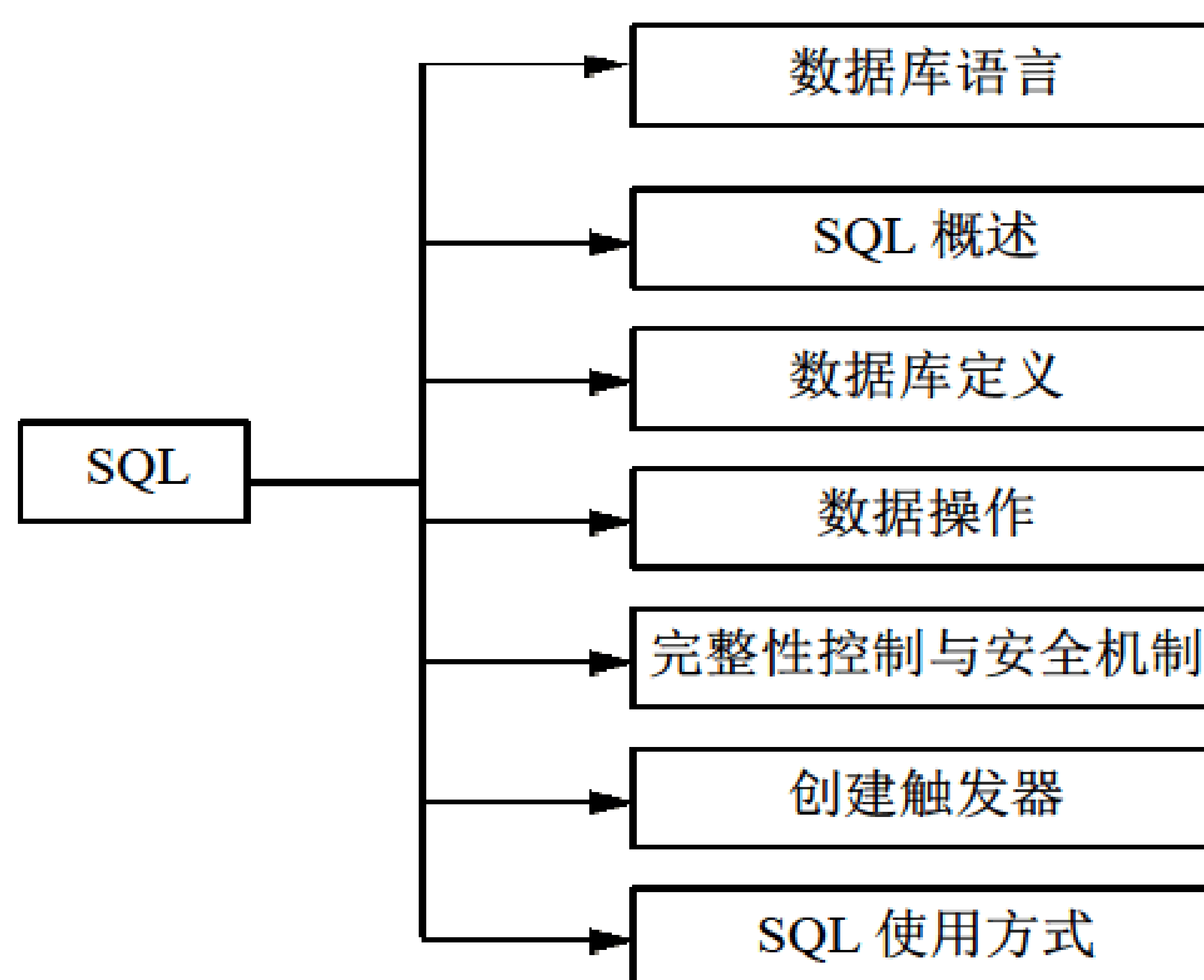


图 5-1　SQL 知识框图

5.1　数据库语言

数据库语言是用来控制数据库的工具，它是用户与数据库交互的“语言”，SQL 已经成为关系数据库的标准语言。

5.1.1　数据库语言的要素

1．知识点提炼

（1）数据库语言

为了允许用户管理和操作数据库，任何数据库系统都必须提供一种与数据库交互的语言，即数据库语言。数据库语言至少需要能够定义和操作数据库，因此必须包括数据定义和数据操作语句。数据库语言与数据库模型（如层次模型、网状模型和关系模型等）密切相关，不同数据库可能使用不同的数据库语言。作为关系数据库语言典范的 SQL 已经成为流行关系数据库的标准语言，它们都必须支持标准 SQL 语言集。

（2）数据库语言的元素

数据库语言必须提供定义数据库的功能，包括数据库的模式、存储结构和存取方式等，因此数据库语言必须包含数据定义语言，即 Data Definition Language，简称 DDL。数据库语言还必须允许用户操作数据库中的数据，即查询数据库中的数据、向数据库中添加数据、从数据库中删除数据、修改数据库中的数据等，所以数据库语言还包含数据操纵语言，即 Data Manipulation Language，简称 DML。总之，数据库语言必须包含 DDL 和 DML。

2．典型例题

【例 5-1】 用来查询数据库中的数据的语言简称为（　　）。

A．DDL　　B．DCL　　C．DML　　D．DAL

【答案】 C

【解析】 数据操纵语言（Data Manipulation Language，DML）提供了查询数据库中数据、向数据库中添加数据、从数据库中删除数据、修改数据库中数据的功能。

5.1.2 数据库语言的使用方式

1．知识点提炼

（1）交互式

一般情况下，数据库管理系统必须提供一个直接与数据库交互的终端，用户只要使用这个终端连接数据库管理系统，然后就可以直接在这个窗口输入需要执行的数据操纵语句，此时数据库管理系统的终端就会将输入语句的执行结果显示在终端上，用户此时可以再输入更多的数据操纵语句，终端将从数据库服务器获得新的执行结果，并显示在终端上，这种使用数据库语言的方式就是交互方式。实际上，结构化查询语言标准 SQL:1999 标准就定义了 SQL 语句必须被支持的绑定方式（执行方式），其中就包含交互式。

（2）嵌入式

嵌入式也是一种非常常用的方式，在这种方式下，数据库语言必须借助于宿主语言才能够执行，而宿主语言就是常见的高级程序设计语言，譬如 C、Fortran、Pascal 等。在编译包含数据库语言的程序时，首先必须使用数据库系统提供的预处理程序对这个程序做处理，将其转换为宿主语言编译器能够识别的语句，然后再由宿主语言编译器来编译，并最后得到可执行程序。当然，如果数据库提供一种经过修改并扩充了对数据库语言支持的宿主语言编译器，则包含数据库语言的程序就可以直接被这种经过修改的编译器编译，并得到可执行程序。实际上，数据库提供的预处理程序和对宿主语言编译器的修改的主要目的就是将用宿主语言与数据库的通信的功能代码插入到程序中，从而可以得到能够与数据库交互并执行数据库语言的程序。

2．典型例题

【例 5-2】 使用嵌入方式执行数据库语言时，必须选择一种宿主语言，但是在编译包含数据库语言的程序之前，必须做预处理，预处理的目的是什么？（　　）

A．删除不符合宿主语言语法要求的数据库语句，保留那些满足要求的语句

B．处理程序中的数据库语句，利用宿主语言添加与数据库通信的功能，并添加调用数据库执行数据库语句的功能，从而实现执行嵌入数据库语句的功能

C．调用数据库终端程序，并将数据库语句在其中执行，将执行结果显示出来

D．给出处理数据库语句的提示，要求用户做相应的修改，从而提高修改的速度

【答案】 B

【解析】 预处理的主要目的就是添加与数据库服务器通信的功能代码，从而实现数据库语句指定的功能，当然，这里添加的代码必须是符合宿主语言语法的代码，这样才能够被宿主语言编译器识别，并通过编译。

5.2　SQL 概述

SQL 是基于关系模型的，已经成为关系数据库语言的标准，它支持关系数据库的创建、维护和对其中数据的管理。

5.2.1　SQL 语句的特征

1．知识点提炼

（1）SQL 语言标准和关系数据库系统对 SQL 的扩展

目前，SQL 主要存在 3 个标准：ANSI SQL、SQL-92 和 SQL-99。ANSI SQL 是支持最为广泛的标准，几乎所有的关系数据库系统都提供对这个基本标准的支持。在 SQL 迅速被众多厂商接受的情况下，美国国家标准局，即 ANSI 颁布了 SQL 语言的美国标准，即 ANSI SQL。随后国际标准组织（ISO）采纳了美国国家标准，并经过修订于 1989 年形成最初的 SQL 语言标准。1992 年 ISO 对原来的标准做了修改，从而产生了 SQL 的 1992 年标准，即 SQL-92，这个标准也称为 SQL2。在 1999 年，ISO 又对 SQL-92 做了修改，主要是为了适应日益复杂的需求，利用面相对象的概念扩充了 SQL-92 支持的纯粹的关系模型，使 SQL 开始支持方法、封装和综合的用户定义数据类型，这就是 SQL 的 1999 年标准，即 SQL-99，这个标准也可以称为 SQL3。SQL-92 标准还定义了遵守该标准的不同级别，即 Entry、Intermediate 和 Full。大部分流行的关系数据库系统还能达到 Entry 级别的要求。在 SQL-99 标准中，则采用了另外一种方法，它规定关系数据库系统要想声明遵守 SQL-99 标准，则必须遵守它所定义的 Core SQL 级别的所有要求，而这个级别包含了 SQL-92 的 Entry 级别的所有要求、Intermediate 和 Full 级别的部分要求，以及几个新的功能特性。

流行的基于 SQL 的关系数据库系统都提供了对 SQL 基本标准的支持，但是为了完成更加复杂和特定的任务，不同厂商生产的关系数据库系统所使用的语言并不是纯粹的 SQL 语言，它们都对 SQL 做了不同的扩充，以实现开发商定义的功能和增强基于 SQL 的功能。这样，每种开发商开发的关系数据库系统所支持的 SQL 都会稍有不同，它们都有自己的 SQL 语言版本。比如，Oracle 使用的是 PL/SQL 语言，Microsoft SQL Server 使用的是 Transact-SQL 语言。

（2）SQL 语句的特征

① SQL 语言是一个功能丰富的语言。SQL 经常被称为是一种“查询”语言，这很不贴切。设计 SQL 的主要目的之一就是检索数据库中的数据，但是它的功能远不至于此，除了能够作为一个查询工具，它还可以用来执行包括数据定义、数据操纵、数据控制和数据

完整性控制等不同类型的管理功能。

② SQL 不是一种计算机程序设计语言，它不需要处理过程的细节，而是一种非过程语言，也称为第四代语言。实际上，可以认为它是一种用来管理关系数据库系统的数据库子语言，并不会处理与数据库毫无关系的程序设计语言应该处理的问题。在使用 SQL 语句时，用户只需要指定要“做什么”，而不需要指定“怎么做”。

③ SQL 是一种非常简洁的语言，易学易用。只要使用几个基本的核心动词（SELECT、CREATE、DROP、ALTER、INSERT、UPDATE、DELETE、GRANT、REVOKE）就可以实现 SQL 的全部核心功能（数据查询、数据定义、数据操纵、数据控制）。

④ SQL 还支持多种使用方式。用户可以利用关系数据库系统的终端通过 SQL 直接与数据库管理系统交互，执行包括数据查询、数据定义、数据操纵和数据控制在内的所有任务；还可以将 SQL 语句嵌入由高级程序设计语言编写的可执行程序中，执行数据库库操纵和查询的各项功能。

2．典型例题

【例 5-3】 下面关于 SQL 语言及其标准的描述不正确的是（　　）。

A．在需求的推动下，ANSI 首先接受了广大厂商对 SQL 的共同认识，制定了美国国家标准；随后，ISO 也在这个标准的基础之上制定了 SQL 标准

B．ISO 的 SQL 标准不断发展，前后经历了 SQL1、SQL2 和 SQL3 等版本，其中 SQL3 的主要改进就是吸纳面向对象的概念

C．SQL 是高级过程语言，用来实现数据查询、数据定义、数据操纵和数据控制方面的功能

D．SQL 可以以多种方式使用，最典型的两种方式就是终端交互方式和嵌入高级程序语言的方式

【答案】 C

【解析】 SQL 是非过程语言，它与高级程序设计语言不同，不需要处理实现某项功能的细节，而只需要说明“做什么”。

5.2.2　SQL 语句的基本成分

1．知识点提炼

（1）SQL 语句的类型

SQL 使用不同的动词来表示需要执行不同的功能，例如，用 SELECT 来表示要执行查询功能，用 CREATE 来表示要执行创建数据库或者表等数据库对象的功能，而用 DELETE 表示要从数据库中删除某条记录的功能，所以也可以根据 SQL 语句使用的不同动词来将它分成几种类型。

- ❑ 数据查询：SELECT
- ❑ 数据定义：CREATE、DROP、ALTER

- ❑ 数据操纵：INSERT、UPDATE、DELETE
- ❑ 数据控制：GRANT、REVOKE

（2）SQL 语句的基本元素

除了前面介绍的用来表示 SQL 语句要求执行的动作类型的动词，SQL 语句还必须包含需要查询、操纵、定义和控制的对象，如基本表、字段、视图、数据库、存储过程等。

SQL 语句必须指定执行这种功能的条件等额外的信息，所以 SQL 语言引入了子句的概念，它们是用不同的 SQL 关键词引出的 SQL 语句的一个部分。常用的子句有用来指定操作/查询条件的 WHERE 字句、用来实现分组查询功能的 GROUP BY 子句、用来执行组搜索条件的 HAVING 子句和用来指定结果排序方式的 ORDER BY 子句等。

另外，SQL 还支持一些基本的函数，用来在字段上执行一些特定的运算，例如用来统计数量的 COUNT 函数、用来获得最大值的 MAX 函数和获得最小值的函数 MIN、用来求和的函数 SUM 等。

2．典型例题

【例 5-4】 下面关于 SQL 语言的描述正确的是（ ）。

A．SELECT 是用来执行查询功能时使用的动词

B．虽然 SQL 不能执行诸如删除、修改、增加等对数据的操纵操作，但是 SQL 仍然是一个功能丰富的语言，它可以支持多种类型的查询操作，还支持分组查询，以及对查询结果的排序

C．不同类型的 SQL 语句可以从它使用的动词来区别，SELECT 是数据操纵的一种动词

D．SQL 不支持数据的安全控制功能，GRANT、REVOKE 等动词都属于数据库管理系统对 SQL 的扩展部分

【答案】 A

【解析】 SQL 能够执行包括删除、修改、增加等操作在内的数据操纵操作，在执行数据查询时，SQL 支持按照分组查询和对查询结果排序的功能。根据 SQL 使用的不同动词，可以区分 SQL 的类型，SELECT 是数据查询的典型动词。SQL 支持安全控制功能，用户可以通过 GRANT、REVOKE 来为数据库用户设置或者取消权限。

5.3 数据库定义

前面已经说明，SQL 语句可以分成数据定义、数据查询、数据操纵和数据控制几类，其中数据定义 SQL 语句就是用来实现数据库定义功能的。用户可以使用数据定义 SQL 语句创建或者删除数据库，创建、删除或者修改基本表，创建、删除或者更新视图等。

5.3.1 创建数据库和表

1．知识点提炼

（1）创建数据库

SQL 定义了创建数据库的语句，但是并没有说明数据库是什么。实际上，在不同的数据库管理系统中，数据库的底层表现形式可能完全不同，但是它们为用户提供的接口却基本相同。主流关系数据库系统，包括 Oracle、MySQL 和 SQL Server 等，都提供了对创建数据库语句的支持，它们使用的基本语法形式也相同，如下所示：

```
CREATE DATABASE <database name>
```

不同数据库系统的数据库名称有不同的约定，但是使用英文字符、数字和下划线组合起来的名称总是合法的。如果需要创建名称为 MyTest 的数据库，那么只要 SQL 语句成为下面的形式即可：

```
CREATE DATABASE MyTest
```

实际上，在不同的数据库系统实现中，还可能支持大量的额外参数，用来指定数据库文件的初始大小、数据库文件的位置、数据库文件的日志文件位置、数据库的初始大小等各种数据库参数。

（2）删除数据库

如果不再需要使用数据库，那么可以删除数据库管理系统中已经存在的数据库，下面就是删除数据库语句的语法形式：

```
DROP DATABASE <database name>
```

如果要删除前面创建的数据库 MyTest，则应该使用下面的语句：

```
DROP DATABASE MyTest
```

（3）创建表

对数据库用户来说，数据库中的表是存储数据的基本单元，用户可以利用 SQL 语句向表中插入基本的数据。使用 CREATE 语句就可以创建表，它的语法形式如下：

```
CREATE TABLE <table name>(
<column name><data type>[column constraint][DEFAULT <default value>]
[,<column name><data type>[column constraint] [DEFAULT <default value>]]…
[,<table constraint>])
```

① 表名称

在这里给出的创建表的语法中，<table name>表示必须指定表的名称，表的命名规则与

数据库的命名规则相似，而且不同数据库实现对表的名称要求不完全相同。在任何数据库系统中，tblTestTable 都是合法的表的名称。

② 字段名称和数据类型

上述语法结构表明，表必须包含一个字段。<column name>表示字段名称是必需的，<data type>表示字段的数据类型也是必需的。数据类型可以是预定义的数据类型，也可以是自定义的数据类型。预定义的数据类型主要有字符串、数字、日期、布尔等，每一种类型还存在各种子类型，下面只列出几个常用的类型。

字符串：CHAR(20)，VARCHAR(30)；

数字类型：INT，NUMERIC(5,2)，DECIMAL(5,2)，FLOAT(6)；

日期类型：DATE，TIME，DATETIME；

布尔类型：BOOLEAN

③ 字段约束

[column constraint]表示可选的字段级完整性约束条件，可以取以下几种。

NULL：字段可以取空值（不需要赋值）；

NOT NULL：字段不可以取空值（必须赋值）；

UNIQUE：字段取值唯一（该字段不允许有重复值）。

后面章节还将给出字段完整性约束的详细说明。

④ 字段默认值

[DEFAULT <default_value>]表示可以为字段选择默认的值，<default value>就是默认值。

⑤ 表的约束

在后续章节中介绍完整性约束。

下面给出一个创建表的例子。

```
CREATE TABLE tblStudent(
ID INT NOT NULL PRIMARY KEY,
Name VARCHAR(64) NOT NULL,
Birthday DATETIME,
Remark VARCHAR(128)
)
```

在 SQL-99 中，除了使用上述语法形式创建表（这里指基本表），还可以使用类似的语法创建全局临时表和局部临时表，因为它们并不常用，所以这里就不详细介绍了。

2．难点分析

（1）数据库名、表名、字段名都是 SQL 中用来代表对象名称的词，不同的数据库系统的实现可能对名称存在不同的要求。常用数据库系统实现支持的命名规则基本相同，它们都要求名称必须以英文字符开始，并且长度不能太长，有的数据库系统限制长度不能超过

30 个字节。一般情况下，名称中只能包含英文字符、数字和下划线，而不允许存在空格、双字节字符以及句号、逗号、回车符、换行符、加号等特殊字符。虽然某些数据库系统实现支持使用汉字作为数据库名或者表名，但是这么做会导致将数据库从一个系统转移到另外一个系统上时遇到困难。

（2）几乎所有的数据库系统都支持字符、字符串、数值、时间、日期、布尔等类型的字段，但是它们使用的数据类型的名称却千差万别，而且极有可能与 SQL 标准不相符，所以在编写 SQL 语句之前，必须确认使用的数据库系统对数据类型方面的说明。同时，在编写 SQL 语句时最好使用一些比较通用的类型，譬如 VARCHAR，CHAR，INT，DATE，DATETIME 等，而不要使用一些很少用的类型，譬如 NUMBER/NUMERIC、BIT、SMALLINT 等。

3．典型例题

【例 5-5】 请根据下面的要求写出创建数据库及创建客户和联系人表的 SQL 语句。

使用某个数据库系统实现的终端连接数据库，创建一个名称为 MyStreet 的数据库，并在其中新建两个表，用来存放客户和联系人的基础数据，客户表包括如下字段：

客户 ID，整型，要求唯一，作为主键

客户名称，字符串，最长 128 字节，不允许为空

客户地址，字符串，最长 64 字节，允许为空

客户邮编，字符串，最长 16 字节，允许为空

客户电话，字符串，最长 16 字节，允许为空

客户信誉值，整型，默认值为 0

客户描述，字符串，最长 255 字节，允许为空

联系人表必须包括如下字段：

联系人 ID，整型，要求唯一，作为主键

联系人姓名，字符串，最长 64 字节，不允许为空

联系人手机，字符串，最长 32 字节，允许为空

联系人描述，字符串，最长 255 字节，允许为空

最后再写出删除这个数据库的 SQL 语句。

【答案】

下面就是这个例题的完整 SQL 语句。

```
CREATE DATABASE MyStreet

CREATE TABLE Customer(
ID INT PRIMARY KEY,
Name VARCHAR(128) NOT NULL,
Address VARCHAR(64) NULL,
PostalCode VARCHAR(16) NULL,
```

```
Telephone VARCHAR(16) NULL,
Credit INT DEFAULT 0,
Remark VARCHAR(255) NULL
)

CREATE TABLE CorrMan(
ID INT PRIMARY KEY,
Name VARCHAR(64) NOT NULL,
Mobile VARCHAR(32) NULL,
Remark VARCHAR(255) NULL
)

DROP DATABASE MyStreet
```

5.3.2　定义数据完整性

1．知识点提炼

（1）数据完整性约束

数据库系统不仅仅用来存储和查询数据，还必须保证数据的完整性，即数据准确性和一致性。如果数据的准确性和一致性受到破坏，那么数据库中的数据就是不可信的。所以，数据完整性是必须得到保证的。SQL 提供了大量用来保证数据完整性的约束，它们可以应用于单个列（字段）、单个表或者多个表。应用于单个字段的约束称为字段约束，应用于单个表的约束称为表约束。

（2）NOT NULL 约束

表中可以存在空值，空值不同于零、空字符或者默认值，它是表示该记录的字段没有被设置任何值，而且没有为该字段指定默认值，它是一种未知的值。而零、空字符或者默认值都是一种特定的值。在定义表时，可以要求不允许某些字段不设置任何值，那就是 NOT NULL 约束。

NOT NULL 约束是应用在字段（列）上的，在创建表时可以为字段指定 NOT NULL 约束。被指定 NOT NULL 约束的列将不允许空值。下面的 SQL 语句对 Name 就施加了 NOT NULL 约束。

```
CREATE TABLE CorrMan(
ID INT PRIMARY KEY,
Name VARCHAR(64) NOT NULL,
Mobile VARCHAR(32) NULL,
Remark VARCHAR(255) NULL
)
```

而 Mobile 和 Remark 字段则没有施加 NOT NULL 约束，允许空值存在。在上述定义下，下面的 INSERT 语句将无法成功执行：

```
INSERT INTO CorrMan(ID, Mobile, Remark) VALUES(1, '13501230900', '')
```

而下面的 INSERT 语句就可以执行成功（在表中不存在任何记录的情况下）：

```
INSERT INTO CorrMan(ID, Name, Remark) VALUES(1, '', '')
```

（3）UNIQUE 约束

UNIQUE 约束也是应用于字段(列)上的约束，它要求字段不能出现重复的值。UNIQUE 既是列约束，也是表约束，所以既可以将约束写在列定义上，也可以将约束写在表约束位置，下面就是一个使用 UNIQUE 列约束的典型例子。

```
CREATE TABLE CorrMan(
ID INT PRIMARY KEY,
Name VARCHAR(64) NOT NULL UNIQUE,
Mobile VARCHAR(32) NULL,
Remark VARCHAR(255) NULL
)
```

如果将 UNIQUE 作为表约束，也可以将约束写在表约束位置，如下所示。

```
CREATE TABLE CorrMan(
ID INT PRIMARY KEY,
Name VARCHAR(64) NOT NULL,
Mobile VARCHAR(32) NULL,
Remark VARCHAR(255) NULL,
UNIQUE(Name)
)
```

（4）PRIMARY KEY 约束

与 UNIQUE 约束相似，PRIMARY KEY 约束也是既可以作为字段约束，又可以作为表约束。被 PRIMARY KEY 约束的字段将作为主键使用，作为主键的字段的值不允许完全相同。如果只定义了一个字段作为主键，那么这个字段的值就不允许出现重复。而如果定义了多个字段作为主键，则它们组合的值不能出现重复。下面就是 PRIMARY KEY 作为字段约束的一个例子。

```
CREATE TABLE CorrMan(
ID INT PRIMARY KEY,
Name VARCHAR(64) NOT NULL UNIQUE,
Mobile VARCHAR(32) NULL,
Remark VARCHAR(255) NULL
)
```

与 UNIQUE 类似，如果将其作为表约束，则可以写成下面的形式。

```
CREATE TABLE CorrMan(
ID INT,
Name VARCHAR(64) NOT NULL UNIQUE,
Mobile VARCHAR(32) NULL,
Remark VARCHAR(255) NULL,
PRIMARY KEY(ID)
)
```

2．难点分析

（1）如果将 UNIQUE 作为表约束来使用，则可以同时指定多个字段，下面就是一个同时指定多个 UNIQUE 的例子。

```
CREATE TABLE CorrMan(
ID INT PRIMARY KEY,
No INT NOT NULL,
Name VARCHAR(64) NOT NULL,
Mobile VARCHAR(32) NULL,
Remark VARCHAR(255) NULL,
UNIQUE(Name, No)
)
```

而如果将 UNIQUE 作为列约束来使用，则必须多次使用 UNIQUE 来约束多个列，下面就是上述例子的等价形式。

```
CREATE TABLE CorrMan(
ID INT PRIMARY KEY,
No INT NOT NULL UNIQUE,
Name VARCHAR(64) NOT NULL UNIQUE,
Mobile VARCHAR(32) NULL,
Remark VARCHAR(255) NULL
)
```

3．典型例题

【例 5-6】 请为例题 5-5 创建的表 Customer 和 CorrMan 之间的关系创建一个表，并写出创建这个关系表的 SQL 语句，要求如下：

关系表的名称是 RelCustomerCorrMan；

联系人只能属于一个客户，客户可以存在多个联系人；

关系表不允许重复记录相同客户与联系人的关系；

关系表中不能出现没有说明特定客户和联系人关系的记录；

需要给出客户与联系人关系的说明信息（用字符串字段存放）。

【答案】

下面就是这个例题的完整 SQL 语句。

```
CREATE TABLE RelCustomerCorrMan(
Customer INT,
CorrMan INT,
Remark VARCHAR(255),
PRIMARY KEY(Customer, CorrMan),
UNIQUE(CorrMan)
)
```

【解析】

为了避免出现记录相同客户和联系人关系的记录，所以将 Customer 和 CorrMan 作为组合主键。而这样定义的另外一个附加的作用就是关系表中的所有记录都不允许 Customer 和 CorrMan 字段为空值，所以所有记录都会说明特定客户和联系人的关系。为 CorrMan 字段定义 UNIQUE 约束将保证关系表中不会出现相同联系人与客户关系，从而保证了联系人只能与一个客户建立关系。字段 Remark 用来存放说明关系的内容。

5.3.3 修改和删除表

1. 知识点提炼

（1）修改表的定义

数据库中的表创建后，还可以利用 ALTER TABLE 语句修改表的定义，下面就是修改表的定义的最基本的语法结构。

```
ALTER TABLE <table name>
ADD [COLUMN] <column name> <data type> [column constraint] [DEFAULT <default value>]
| ALTER [COLUMN] <column name> {SET DEFAULT <default value>|DROP DEFAULT}
| DROP [COLUMN] <column name>
```

可以看出，使用 ALTER TABLE 语句可以在已经创建的表<table name>中添加字段、修改字段、删除字段等。其中，执行添加字段的操作时，对添加的字段的描述形式与创建表时完全相同，下面就是一个正确的例子：

```
ALTER TABLE MyTest ADD COLUMN TestMemo VARCHAR(128)
```

与添加字段不同，修改操作则局限于对默认值的修改上，即或者重新设置默认值，或者取消默认值的设置，下面就是一个重新设置默认值的例子：

```
ALTER TABLE MyTest ALTER COLUMN Name SET DEFAULT 'abcdef'
```

但是有些数据库系统的实现就不存在这样的限制，仍然可以按照新增字段的形式重新定义已经存在的字典，而某些数据库系统的实现可能不支持上面给出的示例。所以，在修改表的定义的时候，最安全的做法就是只增加字段或者删除字段，而不修改字段的定义或者默认值，这就需要前期设计的考虑。

ALTER TABLE 可以用来删除不需要的字段，下面就是一个例子：

```
ALTER TABLE MyTest DROP COLUMN Name
```

删除字段将导致字段的定义和字段对应的数据全部被删除。

（2）删除表

如果某些表已经不再需要，那么可以使用 DROP TABLE 语句来删除，下面就是它的简单的语法结构：

```
DROP TABLE <table name>
```

只需要在上述语法结构中将<table name>替换为实际的表的名称就可以删除这表，下面就是删除一个测试表的例子：

```
DROP TABLE MyTest
```

删除表将导致表的定义和表中存储的数据都被删除。

2．难点分析

删除表将导致表的定义和表的数据同时被删除，如果仅仅想清空表中的数据，而不想删除表的定义，那么应该使用 DELETE 语句，而不是 DROP 语句。删除字段也将导致字段的定义和所有记录对应字段的数据都会被删除。

3．典型例题

【例 5-7】 前面定义了一个用来描述客户和联系人关系的表 RelCustomerCorrMan，但是没有定义用来描述关系类型的字段，这里需要修改这个表的定义，添加一个整型字段 Relation，用来保存关系类型，请给出实现这个功能的 SQL 语句。

【答案】

下面就是这个例题的完整 SQL 语句：

```
ALTER TABLE RelCustomerCorrMan ADD Relation INT DEFAULT 0
```

5.3.4　定义和删除索引

1．知识点提炼

（1）创建索引

利用 CREATE INDEX 可以为表创建索引，从而提高检索的效率。下面就是 CREATE INDEX 的语法：

```
CREATE [UNIQUE] [CLUSTER] INDEX <index_name>
ON <table_name>(<column_name> [ASC|DESC] [,<column_name> [ASC|DESC]]…)
```

其中，UNIQUE 是可选的参数，如果使用这个参数，那么将导致建立一个取值唯一的索引，即在向表<table_name>插入新的记录时，数据库管理系统将检测索引字段中是否存在相同的值，如果存在，则不允许插入字段值相同的记录；CLUSTER 也是可选参数，如果使用这个参数，那么表明要建立的索引是聚簇索引，即索引项的顺序与表中记录的物理顺序是一致的。

<index_name>是索引的名称，<table_name>是要为其建立索引的表的名称，<column_name>用来指定作为索引的字段的名称，ASC 或者 DESC 用来指定索引的顺序，ASC 表示使用升序排列，而 DESC 表示使用降序排列。

这里给出一个创建索引的例子，它可以用来在表 Customer 上为 Name 字段建立升序索引 IndexCustomerName，下面就是建立索引的 SQL 语句：

```
CREATE INDEX IndexCustomerName ON Customer(Name ASC)
```

（2）删除索引

利用 DROP INDEX 可以删除索引，其语法形式非常简单，如下所示：

```
DROP INDEX <index_name>
```

下面就是删除前面创建的索引 IndexCustomerName 的 SQL 语句：

```
DROP INDEX IndexCustomerName
```

2．难点分析

① 索引可以用来加快检索速度，例如前面为 Name 字段创建了索引，则在执行下面的 SQL 语句时，数据库系统将使用这个索引来迅速定位记录，并给出查询结果：

```
SELECT * FROM Customer WHERE Name='jrays'
```

对字符串类型的字段建立索引时需要注意的问题是，建立索引并不一定能够提高检索效率，如果不需要完全匹配字段的内容，那么对字段建立索引并不会改善效率，前面创建的索引 IndexCustomerName 对下面的 SELECT 语句不会有任何帮助：

```
SELECT * FROM Customer WHERE Name LIKE '%jrays%'
```

② 除了可以提高检索效率，索引还存在其他很多作用，如下：

- ❑ 索引可以大大加快数据检索速度，提高检索效率；
- ❑ 通过创建唯一索引，可以保证数据记录的唯一性；
- ❑ 索引可以加速表与表之间的连接，在实现数据的参照完整性方面有特别重要的意义；

- 在使用 ORDER BY 和 GROUP BY 子句进行数据检索时，索引可以显著减少查询中分组和排序的时间；
- 使用索引可以在检索数据的过程中允许数据库系统使用优化器，提高了系统性能。

3．典型例题

【例 5-8】请编写 SQL 语句，为前面定义的客户和联系人关系表 RelCustomerCorrMan 创建两个索引，索引字段是 Customer 和 CorrMan。

【答案】

下面就是需要编写的 SQL 语句：

```
CREATE INDEX idxCustomer ON RelCustomerCorrMan(Customer ASC)
CREATE INDEX idxCorrMan ON RelCustomerCorrMan(CorrMan ASC)
```

5.3.5　定义和删除视图及可更新视图

1．知识点提炼

（1）视图概念

前面章节中使用 CREATE TABLE 定义的表也称为基表，或者基本表，它是与虚表相对的概念。基表是用来描述数据库中数据存储逻辑的定义，而视图则是一个虚表，它只是定义了利用来自一个或者多个基表中的数据形成一个新的虚拟表的查询，而在使用视图时，可以与使用基表的方式相同，即可以从视图中检索出需要的数据。所以，如果需要从多个表中检索数据，或者只需要检索表中的某一部分数据，那么为其特殊定义的视图可能比基表更加方便，而视图存在的目的就是这个。

在数据库中，视图实际上只是存在定义的一个对象，而并不存在具体的数据，通过视图检索出来的数据都是在检索的时候从基表中按照视图的定义提取出来的。定义大量的视图并不会增加数据库的物理数据的尺寸，但是却可以大大简化执行复杂查询的 SQL 语句。

（2）定义视图

视图必须与查询语句密切结合，实际上，视图就是一个查询结果集，所以在定义视图时必须指定用来生成视图的查询语句，下面就是定义视图的 SQL 语句的语法：

```
CREATE VIEW <view name> [(<view column names>)]
AS <SELECT Statement>
```

<view name>是需要创建的视图的名称，<view column names>是视图中列的名称，<SELECT Statement>是用来生成视图的查询语句。查询语句的结果的列就是创建的视图的列。下面就是一个用来生成基表中某些数据的直接快照的典型的例子。

```
CREATE VIEW MyViewTestCustomer(ID, Name, Department)
AS SELECT ID, Name, Department FROM Customer WHERE ID>10000
```

视图最主要的用途之一就是使用连接查询来连接多个表，从而形成多个表的组合视图，下面就是一个典型的例子。

```
CREATE VIEW MyViewTestCustomer(ID, Name, Department)
AS SELECT ID, Name, Department.Name FROM Customer LEFT JOIN Department ON
Customer.Department=Department.ID WHERE ID>10000
```

上面的查询语句使用了连接查询语句，还可以使用复杂的 WHERE 子句来实现类似的功能，下面就是一个例子。

```
CREATE VIEW MyViewTestCustomer(ID, Name, Department)
AS SELECT ID, Name, Department.Name FROM Customer, Department WHERE ID>10000
AND Customer.Department=Department.ID
```

（3）删除视图

从数据库中删除视图的语法形式非常简单，下面就是完整的语法形式：

```
DROP VIEW <view name>
```

删除视图时必须指定要删除的视图的名称，下面就是删除前面创建的视图的 SQL 语句：

```
DROP VIEW MyViewTestCustomer
```

从数据库中删除了视图 MyViewTestCustomer 后，就可以重新创建同名的视图。当然，由于视图只是数据库数据的一个快照，删除视图对数据库中的数据没有任何影响。

（4）可更新视图

视图是数据库中基表数据的一个快照，在某些情况下，视图就是表的直接映射，那么就可以用来修改数据库中的数据，即可更新视图。例如，下面的视图就可以用来更新数据。

```
CREATE VIEW MyViewTestCustomer(ID, Name, Department)
AS SELECT ID, Name, Department FROM Customer WHERE ID>10000
```

而如果视图中的字段不直接对应基表的字段，那么就不能用来更新数据，下面就是创建不可更新的视图的 SQL 语句。

```
CREATE VIEW MyViewTest(Num, TotalSalary, AverageSalary)
AS SELECT COUNT(ID), SUM(Salary), AVG(Salary) FROM Employees
```

实际上，MyViewTest 不是可更新视图的原因非常简单，就是视图的字段都利用了类似 COUNT、SUM 和 AVG 这样的汇总函数了，视图中的记录不能一一对应到基表中的记录上。

2．难点分析

① 在定义新的视图时，必须提供名称，而不一定需要提供视图的列的名称，只有在以下两种情况中，才必须提供视图的列的名称。

- ❑ 视图列的值是由基表的列（字段）的值经过计算后得到的值，而不是直接使用基表中列的值；
- ❑ 如果视图使用了连接查询语句，连接多个表，其结果中包含来自多个表的列的名称相同。

当然，在某些情况下，即使不必须提供列名，为了自己的需要，提供列名也是可以的。在某些情形下，提供列名只是为了重命名字段，下面就是一个例子。

```
CREATE VIEW MyViewTestCustomer(ID, Name, DepartmentID)
AS SELECT ID, Name, Department FROM Customer WHERE ID>10000
```

如果不需要重新命名基表的字段名称，那么可以不指出视图的列的名称，使用下面的SQL 语句就可以创建这个视图。

```
CREATE VIEW MyViewTestCustomer
AS SELECT ID, Name, Department FROM Customer WHERE ID>10000
```

② 可更新视图要求视图的字段（列）必须来自可更新的源表的列，而视图的记录必须与单个基表中的记录一一对应，否则视图就是不可更新的。如果利用了汇总、聚集功能函数，那么视图也是不可更新的。

3．典型例题

【例 5-9】 在前面章节的例题中，已经创建了 Customer 和 CorrMan 表，用来存放客户和联系人数据，还创建了用来描述客户和联系人关系的表 RelCustomerCorrMan。请编写SQL 语句，为上述 3 个表创建视图，用来给出存在关系的客户和联系人的名称，要求如下：

① RelCustomerCorrMan 表中每一条记录在视图中对应一条记录；

② 视图中必须直接给出客户的名称和联系人的名称；

③ 视图中必须给出客户和联系人的 ID 字段的值。

【答案】

下面就是需要编写的 SQL 语句：

```
CREATE VIEW RelCustomerCorrManView(CustomerID, CustomerName, CorrManID,
CorrManName)
AS SELECT Customer.ID, Customer.Name, CorrMan.ID, CorrMan.Name FROM RelCus-
tomerCorrMan LEFT JOIN Customer ON RelCustomerCorrMan.Customer=Customer.ID
LEFT JOIN CorrMan ON RelCustomerCorrMan.CorrMan=CorrMan.ID
```

5.4　数据操作

在定义数据库（基表和视图）后，就可以执行各种数据操作，包括查询数据库中的数据、增加和修改数据库中的数据。

5.4.1 SELECT 语句的基本结构、简单查询、选择、投影

1．知识点提炼

（1）SELECT 语句的基本语法结构

SELECT 语句用于执行查询操作，可以从数据库中的一个或者多个表中查询满足特定要求的数据，下面是它的基本语法结构。

```
SELECT[DISTINCT|ALL] {*|<field list>} FROM <table>[{,<table>}…]
[WHERE <search condition>]
[GROUP BY <grouping specification>]
[HAVING <search condition>]
[ORDER BY <order condition>]
```

可以看出，SELECT 语句必须包含 SELECT 子句和 FROM 子句，而其他所有子句都是可选的。上述 SELECT 语句包含的子句是按照如下顺序被处理的：

FROM 子句；

WHERE 子句（可选）；

GROUP BY 子句（可选）；

HAVING 子句（可选）；

SELECT 子句；

ORDER BY 子句（可选）。

在编写 SQL 语句时，如果需要包含多个子句，则必须按照上述语法结构的顺序给出。

（2）SELECT 子句

在前面给出的基本语法结构中，SELECT 子句包含两个可选的关键字 DISTINCT 和 ALL。如果使用关键字 DISTINCT，则在返回的结果中将删除相同的行，而如果使用 ALL 关键字，则将在结果中保留所有行，而不会删除相同的行。这里给出一个简单的例子，在前面创建的用来描述客户和联系人关系的 RelCustomerCorrMan 表中，由于一个客户可以存在多个联系人，因此表中可能存在同一个客户的多条记录，而如果仅仅需要查询表 RelCustomerCorrMan 中涉及的所有客户的 ID，则可以使用 DISTINCT 关键字。下面就是这个简单的 SELECT 语句：

```
SELECT DISTINCT Customer FROM RelCustomerCorrMan
```

当然，也可以要求返回所有行，不排除相同的行，那么可以使用 ALL 关键字，下面就是使用这个关键字的 SQL 语句：

```
SELECT ALL Customer FROM RelCustomerCorrMan
```

在默认情况下，SELECT 子句使用 ALL 关键字，所以，如果需要使用 ALL 关键字，

那么就可以不指定关键字，即下面的 SELECT 语句与前面这个语句功能相同：

```
SELECT Customer FROM RelCustomerCorrMan
```

除了 DISTINCT 和 ALL 可选项，SELECT 子句必须包含一个用来组成返回结果列的列表，或者使用星号来要求给出所有源表中的列。如果没有指定星号，那么就必须按照每一列从源表列中派生的次序指出每一列，下面就是指定列的语法形式：

```
SELECT [DISTINCT|ALL] <derived column> [[AS] <column name>]
[{, <derived name> [[AS] <column name>]}…]
```

从上面语法结构可以看出，如果需要查询结果包含多个列，那么需要用逗号将其隔开。在大多数情况下，<derived name>就是源表中的列，在其他一些情况下，<derived name>是包含源表中列的表达式；<column name>是用来重新命名的查询结果中的列的名称，如果不给出<column name>，则查询结果中列的名称与源表中列的名称相同。下面给出两种情况下的查询的列的定义，第一种就是使用源表中的列的值作为查询结果中的列，下面就是这个例子的 SQL 语句：

```
SELECT Customer AS CusomerID FROM RelCustomerCorrMan
```

另外一种情况就是使用包含源表中列的表达式来作为查询结果中的列，并重新命名这些列，下面就是一个例子：

```
SELECT SUM(Salary) AS TotalSalary, ID AS Employee FROM Employees WHERE ID=1
```

在第一种情况下，如果不重新命名列，那么查询结果中的列的名称就与源表中的列的名称相同。

（3）FROM 子句

FROM 子句用来指定执行查询的数据来源，它可以包含一个或者多个表，还可以包含一个或者多个引用，下面就是 FROM 子句的语法形式：

```
FROM <table reference>
```

最简单的子句就是只包含一个表的 FROM 子句，它与最简单的 SELECT 子句组合，形成如下所示的最简单的查询语句：

```
SELECT * FROM Customer
```

这个查询语句的执行结果就是返回表 Customer 的所有记录（包含所有列，并且自然继承表 Customer 的列名）。如果只需要指定返回其中的两列 ID 和 Name，则可以使用下面的查询语句：

```
SELECT ID, Name FROM Customer
```

这条查询语句的结果也将继承表的列名，如果需要修改查询结果的列名，则可以使用 AS 关键字来实现，下面就是重命名列 ID 的一个例子：

```
SELECT ID AS No, Name FROM Customer
```

（4）WHERE 子句

WHERE 子句相当于 FROM 子句的筛选器，它将按照用户在 WHERE 子句中指定的条件来筛选由 FROM 子句指定的源表中的记录经过计算（多个表的笛卡尔积运算，或者子查询等）得到的记录，在筛选时，每一行都要根据搜索条件进行评估，如果通过评估（即符合 WHERE 子句指定的表达式要求的条件），那么它们将成为查询结果的一行，否则将被丢弃。下面就是 SELECT 子句的语法形式：

```
WHERE <search condition>
```

表达式<search condition>是由一个或者多个谓词组成的，如果<search condition>包含多个谓词，则谓词中间必须包含用来连接多个谓词的逻辑运算关键字 AND 或者 OR。谓词是用来测试可能返回的结果的，例如下面就是只包含一个谓词的 WHERE 子句：

```
WHERE ID>3
```

在 WHERE 子句中，除了可以使用大于号来形成选择谓词，还可以使用大量其他运算符，如表 5-1 所示的就是所有可能使用的 5 类运算符。

表 5-1　WHERE 子句包含的运算符

类　别	运　算　符	含　义
算术运算符	>	大于，例如 Customer.ID > 2
	>=	大于或者等于，例如 Customer.ID >= 2
	<	小于，例如 Customer.ID < 2
	<=	小于或者等于，例如 Customer.ID <= 2
	=	等于，例如 Customer.ID = 2
	<>	不等于，例如 Customer.ID <> 2
集合运算符	IN	包含，例如 Customer.ID IN (1, 2, 3)
	NOT IN	不包含，例如 Customer.ID NOT IN (1, 2, 3)
字符串运算符	LIKE	相似（左相似，右相似等）， 例如 Customer.Name LIKE '%bc'， 或者 Customer.Name LIKE '%bc%'
空值运算符	IS NULL	是空值，例如 Customer.Name IS NULL
	IS NOT NULL	不是空值，例如 Customer.ID IS NOT NULL
逻辑运算符	AND	并且，例如(Customer.Name IS NULL) AND (Customer.ID < 1)
	OR	或者，例如(Customer.Name IS NULL) OR (Customer.ID < 1)
	NOT	非，例如 NOT (Customer.Name IS NULL)

从表 5-1 可以看出，多个谓词可以使用 AND 或者 OR 逻辑运算符来连接，而且可以使用括号来组成由多个谓词组成的表达式，下面就是一个复杂的表达式：

```
((Customer.ID=10) OR(Customer.ID IN(1, 2, 3, 4, 5)))
AND(Customer.Name>'a')
```

只有 ID 字段的值是 10、1、2、3、4 或者 5，并且 Name 字段大于字符串'a'的记录才符合上面表达式的要求。这样就可以组成复杂的 SELECT 语句，如下所示。

```
SELECT ID, Name FROM Customer
WHERE((Customer.ID=10) OR(Customer.ID IN(1, 2, 3, 4, 5)))
AND(Customer.Name >'a')
```

2．难点分析

① 如果要从多个表中查询结果，那么可以在 FROM 子句中指定多个表，下面就是一个从多个表查询结果的例子。

```
SELECT Customer.ID AS ID, Customer.Name AS Name, CustomerType.Name AS Type
FROM Customer, CustomerType WHERE Customer.Type=CustomerType.ID
```

这条语句将从表 Customer 中获得 ID 和 Name 字段，并通过表 CustomerType 中的 Name 字段形成最终查询的结果，表 Customer 和 CustomerType 中的记录（行）按照 Customer.Type = CustomerType.ID 的条件进行连接操作。实际上，在执行 SQL 语句过程中，首先被扫描并执行的是 FROM 子句，如果 FROM 子句包含两个或者多个表，则两个或者多个表中的笛卡尔积将作为初步结果，然后才会扫描 WHERE 子句，来判断笛卡尔积中哪些结果应该作为最终返回的结果。举一个例子，如图 5-2 所示的是 Customer 和 CustomerType 的内容，它们的笛卡尔积的结果如图 5-3 所示，对如图 5-3 所示的两个表的笛卡尔积按照 WHERE 子句指定的条件进行过滤，过滤结果如图 5-4 所示。而 SELECT 子句则是用来指定结果显示形式的，由于示例的 SELECT 子句只指定了 3 个列，而且都进行了重命名，则最终的查询结果应该如图 5-5 所示。

ID	Name	Type	Remark
1	客户 1	1	NULL
2	客户 2	2	NULL
3	客户 3	2	NULL

（a）表 Customer 的内容

ID	Name	Remark
1	类型 1	NULL
2	类型 2	NULL

（b）表 CustomerType 的内容

图 5-2　表 Customer 和 CustomerType 的内容

② WHERE 子句中的算术运算符除了可以应用于数值类型的字段，还可以应用于字符串类型的字段。例如，下面的谓词也是合法的（Name 字段是字符串型字段）：

```
WHERE Customer.Name>'d'
```

Customer .ID	Customer .Name	Customer .Type	Customer .Remark	CustomerType .ID	CustomerType .Name	CustomerType .Remark
1	客户 1	1	NULL	1	类型 1	NULL
2	客户 2	2	NULL	1	类型 1	NULL
3	客户 3	2	NULL	1	类型 1	NULL
1	客户 1	1	NULL	2	类型 2	NULL
2	客户 2	2	NULL	2	类型 2	NULL
3	客户 3	2	NULL	2	类型 2	NULL

图 5-3　表 Customer 和 CustomerType 的笛卡尔积

Customer .ID	Customer .Name	Customer .Type	Customer .Remark	CustomerType .ID	CustomerType .Name	CustomerType .Remark
1	客户 1	1	NULL	1	类型 1	NULL
2	客户 2	2	NULL	2	类型 2	NULL
3	客户 3	2	NULL	2	类型 2	NULL

图 5-4　对笛卡尔积过滤的结果

ID(Customer.ID)	Name（Customer.Name）	Type（CustomerType.Name）
1	客户 1	类型 1
2	客户 2	类型 2
3	客户 3	类型 2

图 5-5　按照 SELECT 子句指定的列给出的查询结果

则 Name 字段的值为'dbc'或者'xt'的记录也符合这个谓词的要求，而 Name 字段的值为'ab'或者'cx'的记录就不符合谓词的要求。除了逻辑运算符 AND、OR、NOT 外，其他运算符都是用来组成单一谓词的，而逻辑运算符是用来连接谓词的。在逻辑运算符中，AND 和 OR 是二元运算符，它们是用来连接两个谓词的，而 NOT 运算符是单元运算符，它们是用来修饰单个谓词的。AND 用来执行“并且”逻辑操作，即只有记录符合由 AND 连接的两个表达式的要求，才满足整个表达式的要求，否则就不满足整个表达式的要求。OR 用来执行“或”逻辑操作，即只要记录符合由 OR 连接的两个表达式中的任何一个表达式，就满足整个表达式的要求。NOT 用来执行取反的逻辑操作，即符合由 NOT 修饰的表达式的记录将不符合最终表达式的要求，不符合由 NOT 修饰的表达式的记录将符合最终表达式的要求。

③ 选择和投影是关系数学最基本的操作。在 SQL 语句中，可以通过 SELECT 语句的 WHERE 子句和 SELECT 子句分别实现选择和投影操作。例如，下面的 SQL 语句：

```
SELECT Name FROM Customer
```

就实现了对表 Customer 的 Name 字段进行投影的功能。SQL 语句还能够同时做多个字段的投影操作，如下所示：

```
SELECT ID, Name FROM Customer
```

除了执行投影操作，SQL 语句还可以执行选择操作，从而只允许某些记录出现在返回的结果中，如下所示的就是一个简单的例子：

```
SELECT * FROM Customer WHERE ID>3
```

当然，在 SQL 语句中，投影和选择操作可以同时进行，只要合并上述 SQL 语句就实现了合并操作的功能，如下所示：

```
SELECT ID, Name FROM Customer WHERE ID>3
```

上面的 SQL 语句将执行对 Customer 表中所有 ID 字段的值大于 3 的记录的 ID 和 Name 两个字段的投影。

3．典型例题

【例 5-10】 假设 Customer 表中包含字段 ID（记录的唯一表示，整型）、Name（名称，字符串类型字段）、Type（类型，整型变量）、Remark（备注，字符串类型），请编写 SQL 语句，为表中满足下面条件的记录的 Name 和 Remark 字段进行投影：

ID 必须大于 0，并且小于 1024；

Name 字段必须不为空（NOT NULL）；

Remark 字段包含字符串“北方客户”。

【答案】

下面就是需要编写的 SQL 语句：

```
SELECT Name, Remark
FROM Customer
WHERE((ID>0)AND(ID<1024))AND(Name IS NOT NULL)AND(Remark LIKE'%北方客户%')
```

5.4.2　字符串比较、涉及空值的比较、日期时间、输出排序

1．知识点提炼

（1）字符串比较

前面已经介绍了 SQL 的所有运算符，除了逻辑运算符，其他所有运算符都可以应用于字符串，只是它们应用于字符串时的含义与应用于其他类型的字段的含义不完全相同。

用来比较数值大小的算术运算符号（>、>=、<、<=、=、<>）都可以应用于字符串类型的字段中，它们将按照字符串包含的字符的 ASCII 码的大小以从左到右的顺序依次进行比较，直到两个字符串出现不同的字符为止，字符的 ASCII 码大的字符串的值也大，例如：

'b'>'a'

'bcccc>='axdfd'

'ab'>'a'

而

'a'<'x'

=和<>则用于比较两个字符串是否完全相同，如果两个字符串完全相同，则使用=进行运算的结果是真，使用<>进行运算的结果是假，而如果两个字符串不完全相同，则使用=进行运算的结果是假，而使用<>进行运算的结果是真。例如，下面表达式的结果都是假：

'abc'='bddfabc'

'abc' <> 'abc'

而下面表达式的结果都是真：

'abcd'='abcd'

'abcd' <> 'abc'

集合运算符 IN 和 NOT IN 也可以应用于字符串类型的字段，下面就是一个使用集合运算符 IN 的典型例子：

```
SELECT * FROM Customer WHERE Name IN('John', 'Rob', 'Mary', 'Lili')
```

上述语句执行的结果是选择了 Customer 表中 Name 字段的值是 John、Rob、Mary 或者 Lili 的记录。而下面的 SQL 语句执行的结果就是选择表 Customer 中 Name 字段的值不是 John、Rob、Mary 或者 Lili 的记录：

```
SELECT * FROM Customer WHERE Name NOT IN('John', 'Rob', 'Mary', 'Lili')
```

在字符串类型的字段上，使用最多的运算符是 LIKE 运算符，这也是字符串类型字段特有的运算符，它将执行字符串匹配操作。与 LIKE 运算符一起，SQL 通过%来指定字符串匹配的方式。例如，下面的 SQL 语句在执行时将进行左匹配的操作：

```
SELECT * FROM Customer WHERE Name LIKE 'John%'
```

即如果 Customer 表中 Name 字段的值的左部分是 John，则该记录将被选择。例如，Name 字段为 John、Johnx、Johnabd 的记录将被选择，而 Name 字段为 Joah、aJohn 或者 abddj 的记录将不被选择。下面的 SQL 语句在执行时将进行右匹配操作：

```
SELECT * FROM Customer WHERE Name LIKE '%John'
```

与前面给出的左匹配的例子相似，Name 字段的值以 John 结尾时，该记录将被选择，否则记录不被选择。

除了左匹配和右匹配，SQL 语句还支持同时匹配查询，下面就是执行同时匹配的 SQL

语句：

```
SELECT * FROM Customer WHERE Name LIKE '%John%'
```

（2）涉及空值的比较

空值是记录的字段没有被设置成任何值时的状况。在 WHERE 子句中，可以用 IS NULL 和 IS NOT NULL 来对字段进行是否空值的过滤。下面就是一个进行空值过滤的 SQL 语句：

```
SELECT ID FROM Customer WHERE Name IS NULL
```

这个 SELECT 将选择那些 Name 字段没有被设置任何值的记录，并对 ID 字段做投影，从而得到 Name 字段为空值的所有记录的 ID 值列表。

下面是一个相似的 SQL 语句：

```
SELECT ID, Name FROM Customer WHERE Name IS NOT NULL
```

这个 SELECT 语句将选择那些 Name 字段设置了特定值（包括内容为空的字符串）的记录，并对 ID 和 Name 字段做投影。如果 Customer 表的内容如图 5-6（a）所示，则上述 SQL 语句生成的结果如图 5-6（b）所示。

ID	Name	Remark
1	名称 1	NULL
2	名称 2	NULL
3		NULL
4	NULL	NULL
5	名称 1	NULL
6	NULL	NULL

（a）

ID	Name
1	名称 1
2	名称 2
3	
5	名称 1

（b）

图 5-6　非空属性值的查询结果

当然，用来过滤字段的内容是否为空值的两个运算符也可以同时运用在 WHERE 子句中，而不同谓词之间使用 AND、OR 连接，下面就是一个例子：

```
SELECT ID, Name FROM Customer WHERE Name IS NOT NULL AND Department IS NULL
```

（3）日期时间

如果表中包含日期时间类型的字符串，那么在编写 SELECT 语句时，也可以将这些字段应用于谓词中而放在 WHERE 子句中，对记录做选择。而在一般的关系数据库系统中，日期时间类型的数据也可以用来进行大小比较，越早的日期越小，下面就是一个用来对日期时间字段进行选择的 SELECT 语句：

```
SELECT ID, Name FROM Customer WHERE UpdatedTime<'2004-12-04'
```

这个 SELECT 语句将用来选择 UpdatedTime 字段的日期早于 2004 年 12 月 04 日的记录，并对 ID 和 Name 字段进行投影。

在某些数据库系统上，SQL 语句中还支持一些与日期时间有关的函数，从而可以实现一些更加方便的功能，CURRENT_TIMESTAMP 就是一个典型的例子，它得到了 SQL Server 和 Oracle 的支持，用来给出当前时间（包括日期），下面就是一个使用此函数的例子：

```
SELECT ID, Name FROM Customer WHERE UpdatedTime<CURRENT_TIMESTAMP
```

这个 SQL 语句将选择表 Customer 中 UpdatedTime 字段保存的日期时间的内容早于数据库系统当前时刻的记录。

（4）输出排序

在 SELECT 语句中，ORDER BY 子句被用来执行输出排序功能，它也是需要处理的最后一个子句。在输出查询结果之前，ORDER BY 将按照指定的字段顺序（升序或者降序）输出最终的查询结果。

在使用 ORDER BY 子句时，只要给出需要对其排序的字段和排序的方式（ASC 用来指定按照升序的方式排序，DESC 用来指定按照降序的方式排序，在默认情况下，使用 ASC 执行升序方式排序），下面就是 ORDER BY 子句的语法格式：

```
ORDER BY <column name>[ASC|DESC] [, <column name> [ASC|DESC]]…
```

显然，ORDER BY 可以用来指定对多个字段执行排序操作。如果指定对多个字段排序时，则按照从左到右的顺序首先给左边的字段排序，然后再为右边的字段排序。例如，下面就是一个包含 ORDER BY 子句的 SELECT 语句：

```
SELECT ID, Name FROM Customer WHERE UpdatedTime<CURRENT_TIMESTAMP
ORDER BY Name ASC, ID DESC
```

在输出这条 SQL 语句查询结果时，ORDER BY 子句首先根据 Name 字段的值对结果排序，只有出现 Name 的内容完全相同时，再根据 ID 字段的值，将 ID 字段值大的记录放在前面，而将 ID 字段值小的记录放在后面，如图 5-7 所示的就是一个可能的排序结果。

ID	Name	ID	Name
3		1	名称 1
5	名称 1	2	名称 2

图 5-7　按照多个字段的属性值为查询结果排序

当前，因为 ORDER BY 使用 ASC 作为默认的排序方式，所以前面 SQL 语句中的 ASC 不是必需的，即也可以使用下面的 SQL 语句：

```
SELECT ID, Name FROM Customer WHERE UpdatedTime<CURRENT_TIMESTAMP
ORDER BY Name, ID DESC
```

除了指定对两个字段排序，还可以对更多的字段执行排序操作，下面就是一个包含对 3 个字段执行排序的 SQL 语句：

```
SELECT ID, Name FROM Customer WHERE UpdatedTime<CURRENT_TIMESTAMP
ORDER BY Name, Department DESC, ID DESC
```

2．难点分析

① 在不同的数据库系统中，SQL 语句中的日期时间数据格式最为复杂。在 SELECT 语句的 WHERE 子句中，当对日期时间字段进行比较时，常量必须满足数据库系统支持的日期时间格式，例如'2004-12-04'就是 SQL Server 数据库系统支持的日期格式，所以前面给出的 SELECT 语句能够在这个数据库系统上执行。在使用日期时间相关的函数时，也必须注意不同数据库系统支持不同日期时间函数的问题。例如，CURRENT_TIMESTAMP 是 SQL Server 和 Oracle 都支持的日期时间函数，而 CURRENT_DATE、LOCALTIMESTAMP 也得到了 Oracle 的支持，CURRENT_TIME 和 LOCALTIME 函数却没有得到 Oracle 的支持。

② 如果不使用 ORDER BY 子句指定排序操作，查询返回的结果将按照表中记录的实际存储顺序输出。

3．典型例题

【例 5-11】 假设 Customer 表中包含字段 ID（记录的唯一表示，整型）、Name（名称，字符串类型字段）、Type（类型，整型变量）、Remark（备注，字符串类型），请编写 SQL 语句，为表中满足下面条件的记录的 Name 和 Remark 字段进行投影：

ID 必须大于 0，并且小于 1024；

Name 字段必须以“李”开始；

UpdatedTime 和 CreatedTime 字段都必须小于 2004-12-04。

【答案】

下面就是需要编写的 SQL 语句：

```
SELECT Name, Remark
FROM Customer
WHERE((ID>0) AND (ID<1024)) AND(Name LIKE '李%') AND (UpdatedTime <'2004-
12-04') AND(CreatedTime<'2004-12-04')
```

5.4.3　多表查询、连接、并、交、差、属性歧义和元组变量

1．知识点提炼

（1）多表查询和基本连接操作

前面给出的 SQL 语句大都是在一个表中执行查询的，而没有考虑多个表之间的关系，但是这种需求是大量存在的。举一个简单的例子，Customer 表中保存了客户信息（包括 ID、名称、地区 ID、类型 ID、创建时间、修改时间、备注等），而在 CustomerType 中保存了客

户的类型信息，在 Area 中保存了区域信息，它们的内容如图 5-8、图 5-9 和图 5-10 所示。

ID	Name	Area	Type	UpdatedTime	CreatedTime	Remark
1	客户 1	1	1	2004-12-04 12:30:00	2004-12-04 12:30:00	NULL
2	客户 2	1	2	2004-12-04 12:35:00	2004-12-04 12:35:00	abc
3	客户 3	2	2	2004-12-04 12:39:00	2004-12-04 12:39:00	NULL

图 5-8　表 Customer 的内容

ID	Name	Remark
1	南方	长江以南地区的客户
2	北京	北京地区的客户
3	其他	其他所有地区的客户

图 5-9　表 Area 的内容

ID	Name	Remark
1	最有价值客户	NULL
2	普通客户	NULL
3	其他客户	NULL

图 5-10　表 CustomerType 的内容

显然，图 5-8、图 5-9 和图 5-10 的表中的数据是存在重要联系的，而且在某些要求中还需要同时显示这几个表中的内容。例如，在某种应用中，需要打印所有客户的 ID、名称、所属区域名称和备注信息，此时就需要同时从 Customer 表和 Area 表中获得数据，即需要执行多表查询。

多表查询最简单的方法就是在 FROM 子句中指定需要执行多表查询的表，在 SELECT 子句中指明查询结果使用的字段的来源，即指明来源于 FROM 子句中多个表中的哪个表。下面就是一个简单的多表查询的例子：

```
SELECT Customer.ID AS ID, Customer.Name AS Name, Area.Name AS AreaName
FROM Customer, Area
```

在执行上述语句时，数据库系统首先对 Customer 和 Area 两个表做笛卡尔积，然后对这个笛卡尔积关系进行 3 个字段（来自 Customer 表的 ID 和 Name 字段，来自 Area 表的 Name 字段）的投影，如图 5-11 所示的就是查询的结果。

ID	Name	AreaName	Remark	ID	Name	AreaName	Remark
1	客户 1	南方	NULL	2	客户 2	其他	abc
1	客户 1	北京	NULL	3	客户 3	南方	NULL
1	客户 1	其他	NULL	3	客户 3	北京	NULL
2	客户 2	南方	abc	3	客户 3	其他	NULL
2	客户 2	北京	abc				

图 5-11　笛卡尔积做投影的结果

可以看出，图 5-11 的结果是毫无意义的，实际上，在做笛卡尔积的时候，必须对结果进行过滤，否则无法得到合适的结果。在上面的示例中，只有当 Customer 表中指定的 Area 的 ID 值与 Area 表的 ID 值相等的行才需要保留，这样就需要修改上述 SQL 语句，如下所示：

```
SELECT Customer.ID AS ID, Customer.Name AS Name, Area.Name AS AreaName
FROM Customer, Area
WHERE Customer.Area=Area.ID
```

实际上，这是最简单的连接操作。

（2）使用元组变量

如果在多个表中存在相同名称的字段，则在引用这些字段时必须指明字段的来源，而如果需要引用具有歧义的字段的数量比较多，并且 SQL 语句本身比较复杂，则 SQL 语句就会变得非常复杂。SQL 语句支持通过元组变量来引用具有歧义的字段。下面的 SQL 语句是直接通过表的名称来引用字段的例子：

```
SELECT Customer.ID AS ID, Customer.Name AS Name, Area.Name AS AreaName
FROM Customer, Area
WHERE Customer.Area=Area.ID
```

在 FROM 子句中可以定义元组变量，并在 SELECT 子句中使用，如下所示的就是上述 SQL 语句的等价语句：

```
SELECT c.ID AS ID, c.Name AS Name, a.Name AS AreaName
FROM Customer c, Area a
WHERE c.Area=a.ID
```

这里需要注意元组变量的定义方式和引用方式。

（3）在两个以上表之间创建连接

前面执行连接查询时，只包含了两个表，而包含两个表以上的连接查询的方式是相似的。如果需要同时显示客户的名称、类型名称和区域名称，那么就需要在表 Customer、CustomerType 和 Area 三个表之间执行连接查询，如下所示：

```
SELECT c.ID AS ID, c.Name AS Name, a.Name AS AreaName, t.Name AS TypeName
FROM Customer c, Area a, CustomerType t
WHERE c.Area=a.ID AND c.Type=t.ID
```

这个 SQL 语句的 WHERE 子句提供了两个 3 表之间的连接条件，同时这里使用了元组变量来避免属性歧义。

（4）使用条件连接（内连接）

前面通过在 FROM 子句中指定多个表来实现连接查询，而连接的条件则是在 WHERE 子句中提供的。SQL 还支持在 FROM 子句中用 JOIN 和 ON 关键字来指定连接条件，从而使 SELECT 语句更加便于理解，下面就是一个例子：

```
SELECT c.ID AS ID, c.Name AS Name, a.Name AS AreaName, t.Name AS TypeName
FROM Customer c JOIN Area a ON c.Area=a.ID JOIN CustomerType t ON c.Type=t.ID
```

上述 SELECT 语句中的 FROM 子句指定的来源是由 3 个表连接而得到的一个抽象的表。除了需要指定表的名称或者元组变量，SELECT 子句与单表查询子句类似。这段 SQL 语句与下面没有使用条件连接的 SQL 语句的功能相同：

```
SELECT c.ID AS ID, c.Name AS Name, a.Name AS AreaName, t.Name AS TypeName
FROM Customer c, Area a, CustomerType t
WHERE c.Area=a.ID AND c.Type=t.ID
```

使用 JOIN 和 ON 关键字实现的条件连接也称为内连接，可以通过使用关键字 INNER 来强行指定连接为内连接的性质，下面就是与前面例子等价的 SQL 语句：

```
SELECT c.ID AS ID, c.Name AS Name, a.Name AS AreaName, t.Name AS TypeName
FROM Customer c INNER JOIN Area a ON c.Area=a.ID INNER JOIN CustomerType t
ON c.Type=t.ID
```

（5）使用外连接

外连接除了允许出现完全匹配的行，还允许出现一些或者全部不匹配的行，这取决于外连接的类型：左连接、右连接和全连接，它们的含义如下。

① LEFT JOIN（左连接）返回所有完全匹配的行和关键字 JOIN 左边表中所有不匹配的行，而在连接结果中，左边表中所有不匹配的行对应的 JOIN 右边表的字段将被置为空值；

② RIGHT JOIN（右连接）返回所有完全匹配的行和关键字 JOIN 右边表中所有不匹配的行，而在连接结果中，右边表中所有不匹配的行对应的 JOIN 左边表的字段将被置为空值；

③ FULL JOIN（全连接）返回所有完全匹配的行和关键字 JOIN 两边表中所有不匹配的行，而在连接结果中，右边表中所有不匹配的行对应的 JOIN 左边表的字段将被置为空值，左边表中所有不匹配的行对应的 JOIN 右边表的字段将被置为空值。

外连接的语法和内连接的语法相同，只是需要指定不同的关键字。内连接使用 INNER JOIN 关键字或者 JOIN 关键字，而外连接使用 LEFT OUTER JOIN、RIGHT OUTER JOIN、FULL OUTER JOIN 关键字或者 LEFT JOIN、RIGHT JOIN、FULL JOIN 关键字（OUTER 关键字是可选的）。

（6）创建自连接

在某些情况下，表需要与自己执行连接操作，前面介绍的多个表的连接的概念也可以应用于表与自己的连接操作中。举一个简单的例子，表 Customer 中存在一个字段 Owner 用来指出这个客户的所有者客户，也就是某个客户可能是另外一个客户的子公司，那么就可以通过 Owner 来描述，如果某个客户存在父公司，那么就在 Owner 中指定。在这种情况下，如果需要查询某个客户的所有父公司的客户信息，那么就需要执行自连接操作，如下就是实现这个功能的 SQL 语句。

```
SELECT cb.ID AS ID, cb.Name AS Name
FROM Customer ca JOIN Customer cb ON ca.Owner=cb.ID
WHERE ca.ID=1
```

容易看出，这里必须使用元组变量，否则无法在 SELECT 子句中引用正确的字段，因为两个源表的名称也完全相同，根本无法通过表来引用正确的字段。

（7）并、交和差

SELECT 语句的查询结果就是一个关系的描述，也可以看作一个集合，因此集合的运算并（UNION）、交（INTERSECT）和差（MINUS）都可以在 SELECT 的查询结果上执行。下面就是执行 UNION 的一个例子。

```
(SELECT ID, Name FROM Customer WHERE ID=1)
UNION
(SELECT cb.ID AS ID, cb.Name AS Name
FROM Customer ca JOIN Customer cb ON ca.Owner=cb.ID
WHERE ca.ID=1)
```

UNION 用于执行并操作，它的结果是合并两个查询的结果，并自动删除重复的结果。所以，上述 SQL 语句执行的结果就是 ID 为 1 的客户和它的所有父公司对应的客户信息。

交和差操作的语法与并操作的语法相似，下面就是执行交操作的例子。

```
(SELECT ID, Name FROM Customer WHERE ID>200)
INTERSECT
(SELECT ID, Name FROM Customer WHERE Name LIKE '李%')
```

执行交操作的结果是两个查询结果的交集，只有既属于第一个查询结果中的记录也属于第二个查询结果中的记录才会在最终的结果中出现。所以，上述 SQL 语句等价于下面的 SQL 语句。

```
SELECT ID, Name
FROM Customer cb
WHERE ca.Name LIKE '李%' AND ID>200
```

差操作的结果是在第一个查询结果中出现，而不在第二个查询结果中出现的记录才会在最终结果中出现。

2．难点分析

① 在执行多表查询时，如果多个表中的字段名称不相同，那么 SELECT 子句就可以按照与单表查询相同的方法来编写，而如果多个表中存在相同名称的字段，则为了避免歧义，在 SELECT 子句需要引用这些字段时，就必须指定源表的名称，并用点号与字段名隔开，即如 Customer.Name 的形式。

② 在执行连接查询时，使用元组变量能够简化 SQL 语句，但是在使用 SQL 语句的元组变量时，必须注意元组变量的名称的处理，过多并且没有规律的元组名称容易导致混乱。

③ 内连接是最常用的条件连接，通过 INNER JOIN 关键字来指定内连接，而内连接是连接的默认类型，因此可以不需要指定关键字 INNER。内连接要求只出现符合 ON 关键字指定的匹配条件的关系（组合记录），不匹配的记录都将被过滤掉。而外连接则允许不完全匹配的记录。

④ 外连接的定义比较复杂，我们可以通过它们执行的效果来理解外连接定义的实际含义。如图 5-12 和图 5-13 是表 Customer 和 Area 的内容，则下面 SQL 语句执行的结果如图 5-14 所示。

```
SELECT c.ID AS ID, c.Name AS Name, a.Name AS AreaName, c.Remark AS Remark
FROM Customer c INNER JOIN Area a ON c.Area=a.ID
```

ID	Name	Area	Type	UpdatedTime	CreatedTime	Remark
1	客户 1	1	1	2004-12-04 12:30:00	2004-12-04 12:30:00	NULL
2	客户 2	2	2	2004-12-04 12:35:00	2004-12-04 12:35:00	abc
3	客户 3	5	2	2004-12-04 12:39:00	2004-12-04 12:39:00	NULL

图 5-12　表 Customer 的内容

ID	Name	Remark
1	南方	长江以南地区的客户
2	北京	北京地区的客户
3	其他	其他所有地区的客户
4	未定义	NULL

图 5-13　表 Area 的内容

ID	Name	AreaName	Remark
1	客户 1	南方	NULL
2	客户 2	北京	abc

图 5-14　内连接的结果

而如果执行外连接，则可以给出 3 种类型的外连接的 SQL 语句，如下所示。

```
SELECT c.ID AS ID, c.Name AS Name, a.Name AS AreaName, c.Remark AS Remark
FROM Customer c LEFT JOIN Area a ON c.Area=a.ID

SELECT c.ID AS ID, c.Name AS Name, a.Name AS AreaName, c.Remark AS Remark
FROM Customer c RIGHT JOIN Area a ON c.Area=a.ID

SELECT c.ID AS ID, c.Name AS Name, a.Name AS AreaName, c.Remark AS Remark
FROM Customer c FULL JOIN Area a ON c.Area=a.ID
```

如图 5-15、图 5-16 和图 5-17 所示的就是上述 3 个 SQL 语句的执行结果。

可以看出，在执行左连接时，Customer 的 Area 字段的值即使在 Area 表中没有对应的记录，“客户 3”也在查询结果中出现了。但是，在右连接时，“客户 3”的记录就没有出现，

而出现了 Area 表中的另外两个没有匹配的记录。在执行全连接时，所有没有出现的记录都出现了。

ID	Name	AreaName	Remark
1	客户 1	南方	NULL
2	客户 2	北京	abc
3	客户 3	NULL	NULL

图 5-15 左连接的结果

ID	Name	AreaName	Remark
1	客户 1	南方	NULL
2	客户 2	北京	abc
NULL	NULL	其他	NULL
NULL	NULL	未定义	NULL

图 5-16 右连接的结果

ID	Name	AreaName	Remark
1	客户 1	南方	NULL
2	客户 2	北京	abc
3	客户 3	NULL	NULL
NULL	NULL	其他	NULL
NULL	NULL	未定义	NULL

图 5-17 全连接的结果

⑤ 并操作基本得到了所有流行关系数据库系统的支持，而交和差则没有得到大部分数据库的支持，所以在使用前必须注意查看数据库系统的相关说明。

⑥ 对两个查询结果做并、交和差等集合运算时，必须注意以下几点：

- ❑ 两个查询语句返回结果必须具有相同数量的列；
- ❑ 两个查询语句返回的结果的相应的列必须具有相同的类型；
- ❑ 结果中重复的行将被自动删除；
- ❑ 第一个查询语句 SELECT 子句中的列名将作为最后返回的结果的列名。

3．典型例题

【例 5-12】 假设 Customer 表中包含客户字段 ID（记录的唯一表示，整型）、Name（名称，字符串类型字段）、Type（类型，整型变量）、Owner（父客户 ID）、Remark（备注，字符串类型），CustomerType 表中保存了客户类型数据，包括与 Customer 表的 Type 字段对应的 ID 字段和客户类型名称的 Name 字段，请编写 SQL 语句，给出满足条件的客户的 ID、名称、类型名称和备注：

ID 必须大于 0，并且小于 1024；

必须存在子客户；

子客户的类型不能为 2。

【答案】

下面就是需要编写的 SQL 语句：

```
SELECT cb.ID AS ID, cb.Name AS Name, t.Name AS TypeName, cb.Remark AS Remark
FROM Customer ca JOIN(Customer cb LEFT JOIN CustomerType t ON cb.Type=t.ID)
ON ca.Owner=cb.ID
WHERE((cb.ID>0) AND(cb.ID<1024)) AND(ca.Type<>2)
```

5.4.4　子查询

1．知识点提炼

（1）子查询

在 SELECT 语句中，可以根据实际的需要来调用一个查询，并使用这个查询的结果。这个被调用的查询就是子查询，而调用子查询的查询称为父查询。子查询还可以调用新的子查询，即进行多层的查询嵌套。

一个子查询可以返回一个简单的数据，也可以返回一个包含多个值的集合，还可以返回一个包含多个列的关系。由于许多数据库系统并不支持返回关系的子查询，因此这里主要介绍前两种子查询。

（2）返回简单数据的子查询

如果子查询只包含一列，而且子查询的结果只有一个结果，那么子查询返回的就是简单的数据，下面就是一个只返回一个简单数据的查询（这里已经假设 Customer 表的 Name 字段不重复）：

```
SELECT ID FROM Customer WHERE Name='abc'
```

这样的查询就可以作为返回简单数据的子查询使用。下面的 SELECT 语句也只能返回一个简单的数据：

```
SELECT MAX(ID) FROM Customer
```

下面给出一个简单的包含子查询的 SELECT 语句：

```
SELECT ID FROM Customer
WHERE Type=(SELECT ID FROM CustomerType WHERE Name='普通客户')
```

而这个包含了子查询的 SQL 语句的功能与下面这个 SQL 语句的功能完全相同：

```
SELECT Customer.ID
FROM Customer JOIN CustomerType ON Customer.Type=CustomerType.ID
WHERE CustomerType.Name='普通客户'
```

（3）返回集合的子查询

与返回简单数据的子查询不相同，也可以使用子查询返回集合，从而可以使用 IN 或者 NOT IN 来使用子查询返回的结果，下面就是一个简单的例子：

```
SELECT ID, Name, Remark FROM Customer
WHERE Type IN(SELECT ID FROM CustomerType)
```

这个嵌套了子查询的 SQL 语句能够选择 Customer 表中 Type 字段的值在 CustomerType 表中出现的记录，并对 ID、Name 和 Remark 进行投影。

2．难点分析

① IN 或者 NOT IN 用来对集合做运算。IN 运算符表示如果集合中包含指定的元素，那么结果为真，否则为假。NOT IN 运算符与 IN 恰好相反。而 IN 可以对返回集合的子查询进行运算，也可以对某一个真正的集合做运算，例如下面就是一个简单的例子：

```
SELECT ID, Name, Remark FROM Customer
WHERE Type IN(1, 2, 3)
```

② 如果子查询返回多个字段，那么就不能使用 IN 或者 NOT IN 集合运算符对子查询结果做处理，而必须使用 EXISTS 或者 NOT EXISTS 运算符来处理，下面就是一个这样的例子：

```
SELECT ID, Name, Remark FROM Customer
WHERE EXISTS(SELECT * FROM CustomerType WHERE ID=Customer.Type)
```

运算符 EXISTS 的含义是，如果子查询返回的结果数不为 0，则运算结果为真，否则为假。NOT EXISTS 与 EXISTS 的含义刚好相反。

前面给出的嵌套子查询的例子也是一个子查询与主查询相关的查询。子查询不能预先处理，而必须在执行主查询时，在过滤每一条记录时，再执行子查询，所以如果 Customer 表中存在大量记录，则子查询也必须执行相应次数，这显然会导致 SQL 的执行效率相当低下。这里给出另外一种实现相同功能的 SQL 语句：

```
SELECT ID, Name, Remark FROM Customer
WHERE Type IN(SELECT ID FROM CustomerType)
```

3．典型例题

【例 5-13】 假设 Customer 表中包含客户字段 ID（记录的唯一表示，整型）、Name（名称，字符串类型字段）、Type（类型，整型变量）、Owner（父客户 ID.、Remark（备注，字符串类型），CustomerType 表中保存了客户类型数据，包括与 Customer 表的 Type 字段对应的 ID 字段和客户类型名称的 Name 字段，请编写 SQL 语句，给出满足条件的客户的 ID、名称、类型名称和备注：

ID 必须大于 0，并且小于 1024；
必须存在子客户；
子客户的类型不能为 2；
必须使用子查询。

【答案】

下面就是需要编写的 SQL 语句：

```
SELECT c.ID AS ID, c.Name AS Name, t.Name AS TypeName, c.Remark AS Remark
FROM Customer c LEFT JOIN CustomerType t ON c.Type=t.ID
WHERE((c.ID>0) AND(c.ID<1024))
AND(c.ID IN(SELECT Owner FROM Customer c1 LEFT JOIN CustomerType t1 ON
c1.Type=t1.ID WHERE t1.ID<>2))
```

5.4.5　插入和修改数据

1．知识点提炼

（1）INSERT 语句

使用 INSERT 语句可以向表中插入新的数据，下面就是 INSERT 语句的语法结构：

```
INSERT INTO <table name>[(<column name>[{, <column name>}…])]
VALUES(<column value>[{, <column value>}…])
```

其中，<table name>用来指定需要向其中插入数据的表的名称，<column name>用来指定字段的名称，<column value>用来指定字段的值。下面就是几个典型的语句：

```
INSERT INTO ActionImportance(ID, Name) VALUES(1, '一般')
INSERT INTO ActionImportance(ID, Name) VALUES(2, '紧急')
INSERT INTO ActionImportance(ID, Name) VALUES(3, '非常紧急')
```

（2）使用 SELECT 语句结果插入记录

如果将 SELECT 语句与 INSERT 语句结合，则可以将 SELECT 语句的结果作为属性值列表插入到表中，下面就是一个例子：

```
INSERT INTO CorrManCity(ID, Name)
SELECT ID, Name FROM CustomerCity WHERE ID>0
```

（3）使用 UPDATE 语句

使用 UPDATE 语句就可以修改数据库中的数据，下面就是 UPDATE 语句的语法结构：

```
UPDATE <table name> SET <new value list>
[WHERE <search condition>]
```

这里的<table name>用来指定需要修改的表的名称，<new value list>用来以“名称=值”的形式给出字段新值。如果需要指定多个字段的新值，那么就需要用逗号进行分隔，下面就是一个例子：

```
UPDATE Customer SET Name='no name', Enabled=1 WHERE ID=1
```

2．难点分析

① 在 INSERT 语句中，在 VALUES 后面指定的属性值的列表必须与表名后面指定的字段列表对应。当然，也可以在表名后面不指定字段名称，而不指定字段名称就与按照定义表时指定的字段列表的顺序指定所有字段的功能是完全相同的。在将 SELECT 语句与 INSERT 语句相结合时，查询语句的结果集的列数必须与目标表中列出的字段数量相同，而且数据类型相对应。

② 使用 UPDATE 语句可以同时指定多个字段的值，这样就导致记录的多个字段的值同时被修改。如果不指定 WHERE 子句，那么表中所有记录的对应字段的值都将被修改，而如果指定 WHERE 子句，那么只有符合 WHERE 子句指定条件的记录的对应字段的值才会被修改。

5.5　完整性控制与安全机制

SQL 支持对数据库中的数据执行表（表约束）甚至字段级别（字段约束或者列约束）的值约束，从而尽可能保持数据的一致性。它还支持对用户（登录）的权限管理，能够为用户赋予表甚至字段级别的读写权限，也可以取消已经赋予的权限，从而保证数据库数据的安全。

5.5.1　主键约束和外键约束

1．知识点提炼

（1）主键约束及其特点

前面章节已经介绍了主键约束，而且已经说明了主键约束（Primary Key）可以作为列约束，也可以作为表约束出现，下面就是两个例子，分别使用使用主键约束作为列约束和表约束，它们的功能是相同的。

```
CREATE TABLE CorrMan(
ID INT PRIMARY KEY,
Name VARCHAR(64) NOT NULL UNIQUE,
Mobile VARCHAR(32) NULL,
Remark VARCHAR(255) NULL
)
```

```
CREATE TABLE CorrMan(
ID INT,
Name VARCHAR(64) NOT NULL UNIQUE,
Mobile VARCHAR(32) NULL,
Remark VARCHAR(255) NULL,
PRIMARY KEY(ID)
)
```

表约束可以非常方便地用来指定由多个字段共同组成的主键约束，而使用列约束就相对比较繁琐。

主键约束与 NOT NULL 约束和 UNIQUE 约束是存在一定关系的，除此之外，主键约束还具有一些其他的特点。下面给出了主键约束的特点。

① 被指定施加主键约束的列（包括以表约束的形式指定的列）将同时自动执行 NOT NULL 约束，即使没有明确在这个列上指定 NOT NULL 约束；

② 被指定施加主键约束的列或者多个列的组合不允许出现重复的属性值，如果施加主键约束的列只有一个列，那么这与 UNIQUE 约束类似，即即使不为单列主键约束施加 UNIQUE 约束，也会自动实施 UNIQUE 约束；

③ 如果需要将主键约束施加在多个列上，那么必须使用表约束的形式，而不能使用列约束的形式，这与 UNIQUE 不同；

④ 表约束是作为表的元素，在定义时与字段（列）定义具有类似的形式。

在一些描述两个表中记录关系的表中，经常需要使用多个字段的组合作为主键。下面给出一个使用主键约束来指定多个字段的组合作为主键的例子。

```
CREATE TABLE CustomerCorrMan(
Customer INT,
CorrMan INT,
Relation INT,
PRIMARY KEY(Customer, CorrMan)
)
```

这就是一个用来描述 Customer 表和 CorrMan 表中记录关系的表，而主键约束自然就包含两个列（不同的 Customer 和不同的 CorrMan 记录）。

（2）外键约束的基本概念

使用关键字 FOREIGN KEY 施加的约束称为外键约束（Foreign Key），也可以称为参照约束。关系数据库中的数据就是需要有目的地链接在一起的，以保证数据的完整性。表之间的这种关系构成了在表之间提供参照完整性的联系，参照完整性的存在之所以必要，就是要保证数据库中数据关联的完整性，即不允许某个表中的数据被修改、更新而对其他表中的数据产生不良影响，或者产生不可预计的错误。在前面举的例子中，Customer（客

户）表中存在一个用来描述客户类型的字段 Type，而其中的值全部引用表 CustomerType（客户类型）表中的记录，在实际情况中，这是非常重要的关系。如果 Customer 的 Type 字段引用的属性值在 CustomerType 表中不存在，那么这个值的含义将无法解释，而且会因为 CustomerType 表中的数据发生变化而产生错误的含义。这就是参照完整性约束，也就是外键约束存在的必要性。数据库系统必须对数据本身的完整性负责任，所以它们都支持基本的外键约束。

外键约束施加在引用其他表中字段的值的表上（表约束），或者引用其他表中字段的值的字段上（字段约束）。

引用其他表中字段的字段需要施加外键约束，这个字段称为引用字段，而这个表也称为引用表。被外键引用的表称为被引用表，被外键字段引用的字段也称为被引用字段。被施加外键约束的字段只允许设置成那些在被引用表中被引用字段中出现的有效的值。

（3）基本的外键约束

外键约束通过 FOREIGN KEY 和 REFERENCES 关键字来实现，它的语法形式比较复杂。如果将外键约束作为列（字段）约束来给出，那么定义列的语法形式如下：

```
<column name> <data type> REFERENCES <referenced table>[(<referenced columns>)][<referential triggered action>]
```

如果作为列约束，那么只需要使用 REFERENCES 关键字就可以实现。可以看出，在使用列约束形式给出外键约束时，可以不给出被引用字段的名称。

如果将外键约束以表约束的形式给出，那么它的语法形式如下：

```
[CONSTRAINT <constraint name>]
FOREIGN KEY(<referencing column> [{, <referencing column>}…])
REFERENCES <referenced table> [(<referenced columns)]
[<referential triggered action>]
```

可以看出，定义外键约束比定义键约束要复杂得多，因为既需要指定引用列，还需要指定被引用表和被引用字段，还需要指定它们之间的对应关系；而且还需要指出在参照完整性被破坏时的处理方式。

定义一个简单的外键约束则比较容易，只需要指定被引用表，其他所有设置都使用默认值，下面就是一个简单的使用外键约束的例子。

```
CREATE TABLE Customer(
ID INT PRIMARY KEY,
Name VARCHAR(128) NOT NULL,
Type INT REFERENCES CustomerType,
Address VARCHAR(64) NULL,
PostalCode VARCHAR(16) NULL,
```

```
Telephone VARCHAR(16) NULL,
Credit INT DEFAULT 0,
Remark VARCHAR(255) NULL
)
```

可以看出，其中的 Type 字段使用了外键约束，因为这里只指定了外键约束的被引用表的名称，而没有指定被引用字段的名称，所以 CustomerType 表的主键字段将被引用。当然，在 SQL 语句中也可以指定被引用的字段，如下所示。

```
CREATE TABLE Customer(
ID INT PRIMARY KEY,
Name VARCHAR(128) NOT NULL,
Type INT REFERENCES CustomerType(ID),
Address VARCHAR(64) NULL,
PostalCode VARCHAR(16) NULL,
Telephone VARCHAR(16) NULL,
Credit INT DEFAULT 0,
Remark VARCHAR(255) NULL
)
```

如果使用表约束的形式修改上述字段约束形式的 SQL 语句，则结果如下所示。

```
CREATE TABLE Customer(
ID INT PRIMARY KEY,
Name VARCHAR(128) NOT NULL,
Type INT,
Address VARCHAR(64) NULL,
PostalCode VARCHAR(16) NULL,
Telephone VARCHAR(16) NULL,
Credit INT DEFAULT 0,
Remark VARCHAR(255) NULL,
FOREIGN KEY(Type)
REFERENCES CustomerType(ID)
)
```

最后两行代码就是指定外键约束的代码。这里还可以指出约束的名称，此时的代码如下所示。

```
CREATE TABLE Customer(
ID INT PRIMARY KEY,
Name VARCHAR(128) NOT NULL,
Type INT,
```

```
Address VARCHAR(64) NULL,
PostalCode VARCHAR(16) NULL,
Telephone VARCHAR(16) NULL,
Credit INT DEFAULT 0,
Remark VARCHAR(255) NULL,
CONSTRAINT FK_CustomerType
FOREIGN KEY(Type)
REFERENCES CustomerType(ID)
)
```

这样，最后 3 行就是指定外键约束的代码，这里给出的约束是表约束。如果指定了上述外键约束，那么 Customer 表中就不允许随便插入 Type 字段为任意值的记录，只有那些被包含在 CustomerType 表中的类型值才允许作为 Type 字段的值，否则操作将失败。

（4）外键约束的冲突处理

前面给出的外键约束的语法结构中，可选项<referential triggered action>是用来描述当参照完整性遭受破坏时的处理方式。当试图修改、删除被引用表中的数据时，如果这种操作可能引起引用字段中出现违反参照完整性要求的情况，则需要采取相应的动作，以弥补这种操作可能导致的数据完整性问题。

下面给出<referential triggered action>的语法结构，以说明它的用法：

```
ON UPDATE <referential action> [ON DELETE <referential action>]
| ON DELETE <referential action> [ON UPDATE <referential action>]
```

可以看出，可以分别指定 UPDATE 或者 DELETE 时的处理方式，而且指定的顺序没有特殊要求。处理方式<referential action>可以有多种，包括 CASCADE、SET NULL、SET DEFAULT、RESTRICT 和 NO ACTION 几种，其中 CASCADE 和 NO ACTION 是得到支持最多的两种方式。CASCADE 处理方式要求进行连带操作，即从被引用表中删除某条记录时，也连带删除引用表中引用这个记录的记录，修改被引用表中的某条记录时，也连带修改引用表中引用这个记录的相关字段的属性值。NO ACTION 是大多数数据库系统的默认的动作，即不进行任何操作，而只使试图修改或者删除被引用表的操作失败。在默认情况下，大多数数据库系统默认使用 NO ACTION 处理方式。下面给出一个使用指定处理方式的外键约束。

```
CREATE TABLE Customer(
ID INT PRIMARY KEY,
Name VARCHAR(128) NOT NULL,
Type INT,
Address VARCHAR(64) NULL,
PostalCode VARCHAR(16) NULL,
```

```
Telephone VARCHAR(16) NULL,
Credit INT DEFAULT 0,
Remark VARCHAR(255) NULL,
CONSTRAINT FK_CustomerType
FOREIGN KEY(Type)
REFERENCES CustomerType(ID)
ON UPDATE CASCADE
ON DELETE NO ACTION
)
```

这样，在修改 CustomerType 字段的 ID 字段的值时，Customer 表的 Type 字段的值也会跟着被修改。而如果要删除 CustomerType 中的某个记录时，必须首先删除 Customer 表中引用这个记录的所有记录，否则删除操作将失败。

2．难点分析

① 主键约束可以用来作为表中记录的度量，即不同记录的主键是不允许相同的。而被施加主键约束的列可能是一个或者多个，如果被施加主键约束的列是多个列，那么只要求这多个列的属性值的组合不重复即可，而不是要求每一个列的属性值都必须不重复。这就是多个列的主键约束与 UNIQUE 约束不相同的地方。如果主键约束只施加一个列上，那么这个列就是记录的度量，它必须不相同，所以施加主键约束就与施加 UNIQUE 基本相同，但是，此时施加主键约束同时就给字段施加了 NOT NULL 约束，而 UNIQUE 不能做到这一点，这又是主键约束与 UNIQUE 约束的差别。

② 外键约束用来描述表之间的参照完整性关系，定义外键约束时，必须遵循以下几个重要的指导原则：

- 被引用字段必须是被引用表中施加 UNIQUE 或者 PRIMARY KEY 约束的字段或者字段组合，最常用的被引用字段就是被引用表中的主键；
- 外键约束可以以表约束或者列约束的形式出现，如果外键约束需要施加在多个列上，那么必须使用表约束的形式定义，而如果外键约束只施加在一个列上，那么可以使用表约束或者列约束的形式给出；
- 引用表中定义外键约束的字段的数量必须与给出的被引用表中的被引用字段的数量相同，而且对每一个引用字段进行配置时使用的数据类型必须与对应的被引用列相同，但是引用字段和被引用字段的名称则不需要一致；
- 如果在定义外键时不指定被引用表中的被引用字段，那么被引用表中的主键将自动作为被引用列；
- 在表上定义外键约束时，必须确认被引用的表已经存在，并且对被引用表的被引用字段已经施加了相应的约束。

需要重点注意的是，被引用列并非必须是主键，只要是被施加 UNIQUE 约束的字段就可以作为被引用字段。

③ 外键约束还可以用来描述多个字段的外键约束。如果某个表需要引用主键包含多个（两个或者两个以上）字段的表中的数据时，就需要使用这种支持多个字段的外键约束。如果在 CorrMan 表中，不同记录要求 Name、Customer 和 Birthday 字段的组合不能相同，那么在引用这个表并作为外键时，就需要使用包含多个字段的外键约束。定义包含多个字段的外键约束与定义只包含一个字段的外键约束的语法相似。

3．典型例题

【例 5-14】 请根据下面的要求创建 Area 表和 Customer 表，并定义它们之间的参照约束。Customer 表包括如下字段：

客户 ID，整型，要求唯一，作为主键；

客户名称，字符串，最长 128 字节，不允许为空；

客户地址，字符串，最长 64 字节，允许为空；

客户邮编，字符串，最长 16 字节，允许为空；

客户电话，字符串，最长 16 字节，允许为空；

客户信誉值，整型，默认值为 0；

客户所属区域，整型，外键，引用 Area 表中的 ID 字段，指定关联修改和关联删除的参照完整性处理方式；

客户描述，字符串，最长 255 字节，允许为空。

Area 表必须包括如下字段：

ID，整型，作为主键；

区域名称，字符串，最长 64 字节，不允许为空。

最后再写出删除这个数据库的 SQL 语句。

【答案】

下面就是这个例题的完整 SQL 语句：

```
CREATE TABLE Area(
ID INT PRIMARY KEY,
Name VARCHAR(64) NOT NULL
)

CREATE TABLE Customer(
ID INT PRIMARY KEY,
Name VARCHAR(128) NOT NULL,
Address VARCHAR(64) NULL,
PostalCode VARCHAR(16) NULL,
Telephone VARCHAR(16) NULL,
Credit INT DEFAULT 0,
Area INT REFERENCES Area ON UPDATE CASCADE ON DELETE CASCADE,
Remark VARCHAR(255) NULL
)
```

5.5.2　属性值上的约束和全局约束

1．知识点提炼

（1）NULL 和 NOT NULL 约束

除了对字段或者表施加约束，还可以对字段的属性值施加约束。前面已经介绍的 NULL 和 NOT NULL 就是对属性值的一种约束，它们要求字段可以取空值，或者不可以取空值。

（2）CHECK 约束

除了与空值相关的约束，CHECK 关键字还可以对字段的值施加更加复杂的约束，这样的约束也称为 CHECK 约束。相对于前面定义的表约束和字段约束，CHECK 约束是一种更加复杂和灵活的约束，它既可以作为字段约束，也可以作为表约束。作为字段约束，CHECK 约束的语法非常简单，如下所示：

```
<column name> <data type> CHECK(<search condition>)
```

作为字段约束的 CHECK 约束，只要求在新增记录、修改字段属性值时必须使最后生效的字段的值满足<search condition>的要求。下面就是一个简单的例子。

```
CREATE TABLE CorrMan(
ID INT PRIMARY KEY,
Sex VARCHAR(4) NOT NULL CHECK(Sex IN('', '男', '女')),
Name VARCHAR(64) NOT NULL UNIQUE,
Mobile VARCHAR(32) NULL,
Remark VARCHAR(255) NULL
)
```

容易看出，这里给出的 CHECK 约束要求 Sex 字段的属性值只能是''、'男'或者'女'，而不允许是其他额外的任何值。

作为表约束，CHECK 约束更加灵活，它可以同时对多个字段提出约束要求，下面就是作为表约束的 CHECK 约束的语法：

```
[CONSTAINT <constraint name>]
CHECK(<search condition>)
```

可以看出，CHECK 的语法结构非常简单，但是<search condition>却非常复杂，几乎可以使用在 WHERE 子句中使用的所有谓词来组织<search condition>。最简单的 CHECK 约束是要求整型字段的值必须在某个范围之内，或者只能取几个固定的值，字符串类型的字段的值必须是一些固定的字符串。下面给出一个稍微复杂的 CHECK 约束的例子。

```
CREATE TABLE CorrMan(
ID INT PRIMARY KEY,
```

```
Sex VARCHAR(4) NOT NULL CHECK(Sex IN('', '男', '女')),
Name VARCHAR(64) NOT NULL UNIQUE,
Mobile VARCHAR(32) NULL,
Remark VARCHAR(255) NULL,
CONSTRAINT cf_ID CHECK(ID>0 AND ID<1024000),
CONSTRAINT cf_Name CHECK(Name <> '')
)
```

（3）创建域及其 CHECK 约束

域是一种与数据类型具有类似功能的概念，它可以看作是一种用户自定义的数据类型。下面就是包含了 CHECK 约束的域定义的语法结构：

```
CREATE DOMAIN <domain name> [AS] <data type> [DEFAULT <default value>]
[CONSTRAINT <constraint name>] CHECK(<search condition>)
```

下面就是一个定义域的例子：

```
CREATE DOMAIN CINT AS INT DEFAULT 0
CONSTRAINT cf_CINT CHECK(VALUE >= 0 AND VALUE<1024000)
```

一旦定义了域，就可以像使用新的数据类型一样使用这个新定义的域，下面就是使用域 CINT 的例子。

```
CREATE TABLE CorrMan(
ID CINT PRIMARY KEY,
Sex VARCHAR(4) NOT NULL CHECK(Sex IN('', '男', '女')),
Name VARCHAR(64) NOT NULL UNIQUE,
Mobile VARCHAR(32) NULL,
Remark VARCHAR(255) NULL
)
```

（4）创建断言及其 CHECK 约束

断言也是一种 CHECK 约束，它与使用域表的 CHECK 约束的唯一差别就是它可以使用多个表。实际上，在数据库中，如果得到支持，它是以一种独立于表的对象存在的，所以创建断言的语句不是包含在表的定义之内，而是与表的定义处于同一个层次。下面就是创建断言的语法结构：

```
CREATE ASSERTION <constraint name> CHECK <search condition>
```

可以看出，除了具有与 CHECK 约束相同的语法结构之外，还多了用来创建断言本身的语法部分 CREATE ASSERTION。从这里容易看出，断言本身就是一种约束，只不过它是一种可以作用于多个表的全局性的约束。下面给出一个创建断言的例子：

```
CREATE ASSERTION af_Customer
CHECK((SELECT COUNT(ID) FROM Customer)<1024000)
```

从前面介绍的子查询可以看出，这条断言使用了一个只返回简单数据的字查询语句的结果，并将其与 1024000 进行比较，作为断言的约束主体。容易理解，如果在数据库中创建了这条断言，那么 Customer 表中就只能保存少于 1024000 条的记录，而向 Customer 中插入第 1024000 条记录将不会成功。

2．难点分析

CHECK 约束的两种最常见的用法就是指定字段的取值范围和指定字段可以取值的值域。下面就是这两种用法的典型例子。

```
CREATE TABLE CorrMan(
ID CINT PRIMARY KEY,
Sex INT CHECK(Sex >= 0 AND Sex<3),
Name VARCHAR(64) NOT NULL UNIQUE,
Mobile VARCHAR(32) NULL,
Remark VARCHAR(255) NULL
)

CREATE TABLE CorrMan(
ID CINT PRIMARY KEY,
Sex INT CHECK(Sex IN(0, 1, 2)),
Name VARCHAR(64) NOT NULL UNIQUE,
Mobile VARCHAR(32) NULL,
Remark VARCHAR(255) NULL
)
```

还可以使用 BETWEEN 和 AND 来组成谓词，下面的 SQL 语句也与上述两个 SQL 语句的功能相同。

```
CREATE TABLE CorrMan(
ID CINT PRIMARY KEY,
Sex INT CHECK(Sex BETWEEN -1 AND 3),
Name VARCHAR(64) NOT NULL UNIQUE,
Mobile VARCHAR(32) NULL,
Remark VARCHAR(255) NULL
)
```

5.5.3　权限、授权、销权

1．知识点提炼

（1）权限

为了保证数据库中数据的安全，数据库系统必须支持用户和角色的概念，不同的用户和角色具有不同的权限，从而保证用户只能查看或者修改被授权的数据，而不能查看或者

修改没有被授权的数据，实现既不干扰正常授权用户的访问，又能够保证数据安全的目的。

在数据库系统中，表、表的列、视图、域、用户定义类型、触发器等对象都可以作为权限管理的对象，可以将这些对象的各种不同的权限赋予不同的用户或者角色。

（2）创建和删除角色

在 SQL 中可以使用 CREATE ROLE 来创建角色，并且使用额外的语句给角色赋予不同的权限，下面就是创建角色的语法结构：

```
CREATE ROLE <role name>
```

要创建一个角色，非常简单，下面就是一个简单的例子：

```
CREATE ROLE Customers
```

删除与创建角色一样简单，其基本语法如下：

```
DROP ROLE <role name>
```

下面就是删除角色的一个例子：

```
DROP ROLE Customers
```

在数据库中创建大量角色，能够方便用户权限的管理。角色实质上就是把具有相同权限的用户聚集在一起，统一管理。

（3）授予权限

使用关键字 GRANT 可以授予角色权限，下面就是它的完整语法格式：

```
GRANT {ALL PRIVILEGES|<privilege list>}
ON <object type> <object name>
TO {PUBLIC|<user or role name>} [, {PUBLIC|<use or role name>}]…
[WITH GRANT OPTION]
```

其中，GRANT 子句包含两个可能的选项，即 ALL PRIVILEGES 和特定的权限列表。如果使用 ALL PRIVILEGES，则当前执行这个 SQL 语句的用户在指定的对象上的所有权限都将被授予指定角色。如果不需要指定所有权限，那么可以使用指定的权限列表，下面就是一个这样的例子：

```
GRANT SELECT ON TABLE Customer TO MyCustomerOperator
```

这个 SQL 语句将赋予用户 MyCustomerOperator 查询表 Customer 的内容。如果需要指定多个权限，那么可以使用逗号进行分隔，例如同时需要指定查询、删除和插入的权限，则可以使用下面的 SQL 语句：

```
GRANT SELECT, INSERT, DELETE ON TABLE Customer TO MyCustomerOperator
```

可以看出，ON 子句用来指定授权的对象，在指出授权用户的名称之前，还应该指定被授权对象的名称。如果需要将表的权限授予角色或者用户，那么就使用 TABLE 类型，而如果要将视图的权限授予角色或者用户，那么就使用 VIEW 类型。

在 GRANT 语句中，<use or role name>用来代指需要为其赋值的角色或者用户，上面的 GRANT 语句已经使用了 TO 关键字来指定授权对象是 MyCustomerOperator。

选项 WITH GRANT OPTION 用来指定是否授予角色或者用户附带授权权限，即如果 GRANT 语句带有这个选项，那么被授权的角色或者用户还可以将这个权限授予第三者；而如果 GRANT 语句不带有这个选项，那么获得授权的角色或用户就没有将这个权限授予第三者的权限。下面就是一个给用户或者角色 MyCustomerOperator 授权，并允许它将通过这个 GRANT 语句获得的权限授予别人的例子：

```
GRANT SELECT, INSERT, DELETE ON TABLE Customer TO MyCustomerOperator
WITH GRANT OPTION
```

在一个 GRANT 语句中，可以同时将相同的权限授予不同的用户或者角色，只要在 TO 子句列表中用逗号隔开多个用户或者角色，下面就是一个例子：

```
GRANT SELECT, INSERT, DELETE ON TABLE Customer
TO MyCustomerOperator, MyOperator1, MyOperator2
WITH GRANT OPTION
```

（4）取消权限

使用 REVOKE 语句可以取消授予用户或者角色的权限，下面首先给出 REVOKE 语句的完整语法结构：

```
REVOKE [GRANT OPTION FOR] {ALL PRIVILEGES|<privilege list>}
ON <object type> <object name>
FROM {PUBLIC|<user or role name>}
{RESTRICT|CASCADE}
```

可以看出，REVOKE 与 GRANT 语句非常相似，而且关键字都互相对应。GRANT OPTION FOR 子句用于取消用户或者角色授予其他用户权限的权限（用户或者角色自己对表或者视图的权限仍然会被保留）。当然，如果在使用 GRANT 语句授予用户或者角色权限时，没有指定 WITH GRANT OPTION 子句，则用户或者角色并不具有赋予其他用户权限的权限，所以不能使用 GRANT OPTION FOR 子句。下面就是取消用户或者角色授予别人权限的权限的 REVOKE 语句：

```
REVOKE GRANT OPTION FOR SELECT, INSERT, DELETE ON TABLE Customer
FROM MyCustomerOperator
CASCADE
```

这里使用了关键字 CASCADE，执行这个语句将导致由用户 MyCustomerOperator 赋给其他用户的对表 Customer 的 SELECT、INSERT 和 DELETE 权限都会自动收回。而如果这里使用 RESTRICT 代替 CASCADE，则用户赋给其他用户的权限不会被收回，下面就是这个 REVOKE 语句：

```
REVOKE GRANT OPTION FOR SELECT, INSERT, DELETE ON TABLE Customer
FROM MyCustomerOperator
RESTRICT
```

REVOKE 语句中的 ON 子句与 GRANT 语句中的 ON 子句的用法完全相同，FROM 子句与 GRANT 语句中的 TO 子句的用法完全相同。

如果在给用户或者角色授权时没有指定 WITH GRANT OPTION 子句，那么就不能指定 GRANT OPTION FOR 子句，下面就是直接取消用户权限的例子：

```
REVOKE SELECT, INSERT, DELETE ON TABLE Customer
FROM MyCustomerOperator
```

（5）授予和取消角色

将角色授予用户，用户将具有角色的所有权限，下面就是为用户授予角色权限的 GRANT 语句的语法：

```
GRANT <role name list>
TO {PUBLIC|<user name list>}
```

下面就是将角色 MyRole 权限授予用户 MyCustomerOperator 的语句：

```
GRANT MyRole TO MyCustomerOperator
```

可以使用 REVOKE 取消授予用户的角色权限，下面就是它的语法结构：

REVOKE <role name list>

FROM {PUBLIC|<user name list>}

下面就是取消上面例子中给 MyCustomerOperator 赋予的角色权限的例子：

```
REVOKE MyRole FROM MyCustomerOperator
```

2．难点分析

① 不同数据库系统对用户和角色的管理方式相差很大。Oracle 支持 CREATE ROLE 和 DROP ROLE 语句，可以使用 SQL 语句来管理角色。而 SQL Server 不支持这两个语句，但是 SQL Server 提供了用来实现类似功能的系统存储过程。下面就是 SQL Server 用来管理角色的几个例子：

```
EXEC sp_addrole 'Managers'

EXEC sp_addrolemember 'Sales_Managers', 'Jeff'
```

② 不同对象支持的权限类型是不相同的。表能够支持 SELECT、INSERT、UPDATE、DELETE、TRIGGER 和 REFERENCES 权限，而视图只能支持 SELECT、INSERT、UPDATE、DELETE、REFERENCES 权限。另外，SELECT、INSERT、UPDATE 和 REFERENCES 权限支持字段级别的权限设置，即可以将这些权限应用在一个或者多个字段，而在其他字段上将其禁用，下面就是一个支持字段级权限的例子：

```
GRANT SELECT(ID, Name) ON TABLE Customer TO MyCustomerOperator
```

执行这条 GRANT 语句将导致用户 MyCustomerOperator 具有查询表 Customer 中 ID 和 Name 字段的权限。

很容易理解，DELETE 和 TRIGGER 不允许支持字段级别权限的设置，这是因为这两个操作不能在字段级别执行，而只能应用于记录上。在编写 GRANT 语句时，权限列表中不能包含不能应用在对象上的权限，例如，下面的 GRANT 语句是不正确的：

```
GRANT DELETE(ID, Name) ON TABLE Customer TO MyCustomerOperator
```

下面的语句也是不正确的：

```
GRANT TRIGGER, SELECT(ID, Name) ON TABLE Customer TO MyCustomerOperator
```

③ 许多数据库系统支持公共角色 PUBLIC，所以登录到数据库上的用户将自动具有 PUBLIC 角色的所有权限，因此，可以通过修改 PUBLIC 角色的权限来允许或者禁止潜在的所有用户的访问。

3．典型例题

【例 5-15】 请在数据库中创建一个角色 MyCustomerRole，将表 Customer 的所有权限都赋予这个角色，然后在将这个角色的权限赋予用户 jrays。

【答案】

下面就是这个例题的完整 SQL 语句。

```
CREATE ROLE MyCustomerRole

GRANT ALL PRIVILEGES ON Customer
TO MyCustomerRole

GRANT MyCustomerRole TO jrays
```

5.6　创建触发器

触发器是在向表中插入记录、从表中删除记录或者修改表中记录的数据时，将自动执行的一段 SQL 语句，是对表中数据执行编辑操作时的关联动作。在流行的数据库系统中，

触发器的使用可以提供一种高效的功能调用实现方式。

1．知识点提炼

（1）触发器

触发器是定义了在修改表的数据的过程中可能需要执行的动作，它同时需要定义执行动作的特征和需要执行的动作。触发器包含的动作由一个或者多个 SQL 语句组成，触发器执行的事件包括修改表的数据、删除表中的记录、插入新的记录到表中等。

（2）创建触发器

创建触发器是通过使用关键字 CREATE TRIGGER 来实现的，下面是创建触发器的完整语法结构。

```
CREATE TRIGGER <trigger name>
{BEFORE|AFTER}
{INSERT|DELETE|UPDATE [OF <column list>]}
ON <table name>
<triggered SQL statement>
```

这是创建触发器最简单的语法形式。其中，<trigger name>是需要创建的触发器的名称，{BEFORE | AFTER}用来指定触发器的执行时刻，{INSERT | DELETE | UPDATE [OF <column list>]}用来指定触发器的类型，<table name>用来指定触发器应用的表，<triggered SQL statement>用来指定触发器的具体内容。下面就是一个典型的例子。

```
CREATE TRIGGER CheckInsertUpdateCustomer
BEFORE INSERT OR UPDATE ON Customer
BEGIN
INSERT INTO Temp VALUES('insert customer/update customer')
END
```

这个触发器在向 Customer 表中插入记录或者修改 Customer 表中的数据时被触发，从而执行触发器的 SQL 语句，如下所示：

```
INSERT INTO Temp VALUES('insert customer/update customer')
```

如果触发器包含多个 SQL 语句，则必须将它们包含在 BEGIN 和 END 中，而如果只包含一个 SQL 语句，那么可以不使用 BEGIN 和 END 关键字，下面就是上述触发器的等价定义方式：

```
CREATE TRIGGER CheckInsertUpdateCustomer
BEFORE INSERT OR UPDATE ON Customer
INSERT INTO Temp VALUES('insert customer/update customer')
```

在定义触发器中，如果使用 BEFORE 关键字，那么这个触发器将在触发触发器的操作

执行之前被执行，而如果使用 AFTER 关键字，那么这个触发器将在操作完成之后被执行。

（3）删除触发器

删除触发器使用 DROP TRIGGER 关键字来实现，下面就是删除前面创建的触发器的例子：

```
DROP TRIGGER CheckInsertUpdateCustomer
```

2．难点分析

① 不同数据库系统支持的创建触发器的 SQL 语句差别很大。譬如在 Oracle 中，如果触发器包含多个 SQL 语句，则必须将它们包含在 BEGIN 和 END 中；而在 SQL Server 中，则必须使用 AS 关键字来引起一个或者多个 SQL 语句，并使用 GO 关键字来表示触发器已经完成。正是基于这个原因，前面没有介绍大量的与触发器相关的 SQL 语句，这里给出不同系统支持的触发器语句。下面就是一个得到 SQL Server 支持的创建复杂触发器的例子。

```
CREATE TRIGGER TR_DeletePowerData_For_Contract ON Contract
WITH ENCRYPTION FOR DELETE AS

DECLARE @ID int
DECLARE data_cursor CURSOR FOR

SELECT ID FROM deleted
OPEN data_cursor
FETCH NEXT FROM data_cursor INTO @ID

WHILE @@FETCH_STATUS=0
BEGIN
DELETE FROM Contract_P WHERE Contract=@ID
DELETE FROM tmpUserRight WHERE TableName='Contract' AND RecordID=@ID
FETCH NEXT FROM data_cursor INTO @ID
END

CLOSE data_cursor
DEALLOCATE data_cursor

GO
```

而下面就是 Oracle 支持的一个复杂触发器的例子。

```
CREATE TRIGGER CheckInsertUpdateCustomer
BEFORE INSERT OR UPDATE OR DELETE
ON Customer
BEGIN
IF TO_CHAR(SYSDATE, 'dy')='星期日' THEN
```

```
RAISE_APPLICATION_ERROR(-20000, 'today is holiday!');
END IF;
END;
```

② 前面定义的触发器都会在合适的时候被触发，但是它们在用户执行某个 SQL 语句过程中，只会被执行一次。而如果用户同时修改多行记录的值，这时需要检查用户修改的每一条记录的数据的合法性，那么就需要在即将修改每一条记录的值时都调用触发器来检查数据合法性，这就需要所谓的行触发器。下面就是一个行触发器的例子。

```
CREATE TRIGGER CheckInsertUpdateCustomer
BEFORE UPDATE ON Customer
FOR EACH ROW
BEGIN
INSERT INTO Temp VALUES('update customer')
END
```

这样在使用 UPDATE 语句同时修改多条记录时，每一条记录的值被修改之前，这个触发器都会被调用一次。

3．典型例题

【例 5-16】请为数据库的 Employee 表创建一个触发器(名称为 CheckEmployeeSalary)，在执行 UPDATE 这个表的 Salary 字段时被触发，它能够在执行操作之前检查插入的字段的值是否满足下面的条件：

修改后的值必须大于原来的值；

修改的增量必须小于 10%。

只有满足这些条件的修改操作才允许执行。分别给出 Oracle 和 SQL Server 支持的创建触发器的 SQL 语句。

【答案】

下面给出 Oracle 支持的创建触发器的 SQL 语句。

```
CREATE TRIGGER CheckEmployeeSalary
BEFORE UPDATE OF Salary
ON Employee
FOR EACH ROW
BEGIN
IF :NEW.Salary <= :OLD.Salary THEN
RAISE_APPLICATION_ERROR(-20000, '没有增长');
ELSEIF :NEW.Salary>:OLD.Salary * 1.1 THEN
RAISE_APPLICATION_ERROR(-20000, '增长过快');
END IF;
END;
```

5.7 SQL 使用方式

SQL 存在两种使用方式，即交互式和嵌入式。交互式 SQL 使用数据库系统提供的终端与数据库服务器建立连接，并以交互的方式直接执行 SQL 语句，同时回显 SQL 语句执行的结果。嵌入式 SQL 用户嵌入到宿主语言中，在程序执行过程中实现执行 SQL 语句的功能。

5.7.1 交互式 SQL

交互式 SQL 就是纯粹的 SQL，用户使用数据库系统的终端与数据库服务器交互，向服务器发送需要执行的 SQL，返回 SQL 语句执行的结果。这是一种使用 SQL 最直接的方式。

5.7.2 嵌入式 SQL

嵌入宿主程序语言的 SQL 可以在执行程序时执行，从而可以结合到宿主程序语言编写的应用程序中。

1．知识点提炼

（1）简单的嵌入式 SQL

将 SQL 语言嵌入到程序中，在使用宿主语言编译程序前，必须做预处理，从而将 SQL 语言转换为能够被宿主程序语言编译器能够识别的代码，最后再做编译操作。而为了让预处理器识别嵌入到程序中的 SQL 语句，就必须要区分宿主程序语言和 SQL 语句。在不同的主程序语言中，区分 SQL 语句的方式不完全相同，在 C、Ada、Pascal 语言中，这是通过 EXEC SQL 来标识的。下面就是一个在 C 语言中嵌入的 SQL 语句的例子：

```
EXEC SQL SELECT ID, Name FROM Customer ;
```

C 语言嵌入的 SQL 语句必须以 EXEC SQL 开始，最终以分号结束。而在 COBOL 语言中，上述语句必须写成下面的形式：

```
EXEC SQL SELECT ID, Name FROM Customer END-EXEC
```

为了描述的方便，下面给出的例子都是使用 C 语言作为宿主语言的。

（2）共享变量

共享变量也称为主变量，它是宿主程序语言向 SQL 语句传递参数的主要方式。在宿主程序中定义共享变量，并为其设置需要传递给 SQL 语句的参数，然后就可以在 SQL 语句中使用。下面就是在 C 语言中声明共享变量的方式。

```
EXEC SQL BEGIN DECLARE SECTION ;
INT id, ind_id, ind_name;
```

```
CHAR name[64], remark[256];
EXEC SQL END DECLARE SECTION ;
```

定义的共享变量可以在 SQL 语句中引用，为了与 SQL 语句中的属性变量区别，在引用时必须在共享变量之前添加冒号，下面就是一个引用共享变量的例子：

```
EXEC SQL SELECT ID, Name FROM Customer WHERE ID=:id OR Name=:name ;
```

还可以将共享变量放在 SELECT 子句中使用，下面就是一个例子：

```
EXEC SQL SELECT ID, Name INTO :id, :name FROM Customer ;
```

（3）游标

SQL 语句的 SELECT 语句返回的结果可能不止一条记录，所以需要使用基于集合的功能，这是通过游标实现的。在主程序语言中，移动游标就可以在 SELECT 语句的所有结果上移动。下面就是定义游标的语法结构：

```
EXEC SQL DECLARE <cursor name> CURSOR FOR <SQL statement>
```

这里的<cursor name>用来指定游标的名称，而<SQL statement>用来指定游标代表的 SELECT 语句。下面就是一个简单的例子：

```
EXEC SQL DECLARE MyCustomerCursor CURSOR FOR
SELECT ID, Name FROM Customer ;
```

这只是一个声明游标的语句，它并不会促使这里指定的查询语句立刻被执行。而还必须调用游标的其他功能才可以执行游标，下面就是执行这个游标定义的查询语句的代码：

```
EXEC SQL OPEN MyCustomerCursor ;
```

而关闭游标的方法也非常简单，下面就是关闭这个游标的例子：

```
EXEC SQL CLOSE MyCustomerCursor ;
```

（4）获得查询结果数据

打开游标后，应用程序就可以按照一定的方法获得查询的执行结果。当然，在使用之前，必须打开游标，下面就是打开游标的代码：

```
EXEC SQL OPEN MyCustomerCursor;
```

游标打开后，就可以使用 FETCH 来获得游标处理的查询结果的当前记录值。而在打开游标时，游标自动定位在第一条结果记录上。下面就是使用 FETCH 关键字获得查询结果当前记录属性值的语法结构：

```
EXEC SQL FETCH FROM <cursor name> INTO <variables> ;
```

这里的<cursor name>用来指定游标的名称，<variables>用来指定共享变量的列表，下面就是一个例子：

```
EXEC SQL FETCH FROM MyCustomerCursor INTO :id, :name ;
```

（5）使用定位 UPDATE 或者 DELETE 语句

使用光标可以定位记录。在光标定位记录时，可以针对当前记录执行 UPDATE 或者 DELETE 操作，下面就是它们的语法结构：

```
EXEC SQL DELETE FROM Customer WHERE CURRENT OF <cusor name> ;
EXEC SQL UPDATE Customer SET Name=:name WHERE CURRENT OF <cusor name> ;
```

下面是使用上述语法结构的例子：

```
EXEC SQL DELETE FROM Customer WHERE CURRENT OF MyCustomerCursor ;
EXEC SQL UPDATE Customer SET Name=:name WHERE CURRENT OF MyCustomerCursor ;
```

（6）动态 SQL

前面编写嵌入 SQL 时，SQL 语句都是固化在程序代码中，一旦程序被编译，这样的 SQL 是不能根据用户的输入或者程序配置参数来修改的。如果需要使用在程序运行过程中可以改变的 SQL 语句，就需要使用动态 SQL 技术，下面就是使用动态 SQL 语句的语法结构：

```
EXEC SQL PREPARE <dynamic sql name> FROM <string variable>
```

这里的<dynamic sql name>用来指定动态 SQL 语句的名称，而<string variable>用来指定包含了完整动态 SQL 语句的共享变量。下面就是一个典型的例子。

```
EXEC SQL BEGIN DECLARE SECTION ;
CHAR sql[256];
CHAR SQLSTATE[6];
EXEC SQL END DECLARE SECTION ;

sprintf(sql, "%s%d", "DELETE FROM Customer WHERE ID=", id);

EXEC SQL PREPARE MyCustomerSQL FROM :sql
```

而执行动态 SQL 语句非常简单，下面就是执行上述动态 SQL 语句 MyCustomerSQL 的代码：

```
EXEC SQL EXECUTE MyCustomerSQL
```

2．难点分析

① 共享变量是用来向 SQL 语句传递参数的，它可以应用在 SQL 语句的各个部分，但是不能作为 SQL 标识符，即不能通过共享变量传递目标名，譬如表名、列名等。

② 如果将检索结果保存到 SQL 语句中，那么出现空值时可能出现问题，为了能够识别返回值是否为空值，那么就需要使用下面的形式来返回检索结果的状态。

```
EXEC SQL SELECT ID, Name INTO :id :ind_id, :name :ind_name
FROM Customer WHERE ID=:id OR Name=:name ;
```

如果检索的结果出现空值，那么对应的 ind 变量将为–1，否则该值为 0。当然，在使用共享变量 ind_id 和 ind_name 之前，必须将它们声明成整型变量。

③ 关闭游标将允许应用程序释放与游标相关的资源，所以在不使用游标时，应该及时关闭游标。

④ 在使用 FETCH 获得查询结果时，游标也自动被移动，从而定位到下一条记录上。

3．典型例题

【例 5-17】 表 Customer 由 ID、Name 和 Enabled 字段构成，使用 C 语言编写程序，在其中嵌入 SQL 语句，对其中的记录做如下处理：

如果 Enabled 字段为 0，则删除这个记录；

如果 Enabled 字段不为 0，而且 Name 字段为空，则将其设置为 no name；

打印每一条记录的内容（包括被删除的和修改前的记录）。

【答案】

```
EXEC SQL BEGIN DECLARE SECTION ;
INT ID, Enabled, ind_id, ind_name, ind_enabled;
CHAR Name[64];
CHAR SQLSTATE[6];
EXEC SQL END DECLARE SECTION ;

EXEC SQL DECLARE MyCustomerCursor CURSOR FOR
SELECT ID, Name, Enabled FROM Customer ;

EXEC SQL OPEN MyCustomerCursor ;

WHILE(1) {

EXEC SQL FETCH FROM MyCustomerCursor INTO :id, :name, :enabled ;
IF(!(STRCMP(SQLSTATE,"02000")))
BREAK;
IF(Enabled == 0)
EXEC SQL DELETE FROM Customer WHERE CURRENT OF MyCustomerCursor ;
ELSE IF(Enabled == 1 && STRCMP(Name, '') == 0)
EXEC SQL UPDATE Customer SET Name='no name'
WHERE CURRENT OF MyCustomerCursor ;
```

```
printf("%d,%s,%d\n", ID, Name, Enabled);

}

EXEC SQL Close MyCustomerCursor ;
```

练习题

一、请根据下面给出的文字和提供的被选答案，选择正确的答案填写在空格处，并将正确答案的编号（A、B、C 或者 D．写在答题纸上。

1．结构化查询语言，简称__①__，它是用来操作__②__的主要手段之一，根据操作的不同类型，可以将它分成几类，包括__③__。下面就是一个典型的结构化查询语句：

```
SELECT A1, A2, A3 FROM T1, T2 WHERE L
```

其中，A1（A2, A3）、T1（T2）和 L 分别是__④__。

① A．SQP　　B．PQP　　C．OPS　　D．SQL

② A．函数　　B．关系数据库

C．数据文件　　D．计算机

③ A．数据查询、数据操纵和数据安全　　B．数据定义、数据操纵和数据控制

C．数据查询、数据操纵和数据编辑　　C．数据定义、数据操纵和数据输入

④ A．目标列名、源表或者视图名、数值表达式

B．目标列名、源表或者视图名、逻辑表达式

C．源表或者视图名、目标列名、逻辑表达式

D．源表或者视图名、目标列名、数值表达式

2．对数据库的数据做合法性和完整性控制是__⑤__的一个主要功能，最典型的控制就是通过所谓的约束定义的数据规则来实现的。从基本表级别来看，约束基本可以两类，它们是__⑥__约束和__⑦__约束，其中，前者可以用来同时定义对多个字段的约束，而后者只能定义对一个字段的约束，如果使用后者而且需要对多个字段做约束，则必须给出多个约束，分别作用在这些字段上。下面给出的就是一个使用了所谓的主键约束和 UNIQUE 约束的例子，其中 UNIQUE 约束是同时加在 Name 和 Birthday 两个字段上的：

```
CREATE TABLE CorrMan(
ID INT __⑧__,
Name VARCHAR(64) NOT NULL,
Birthday VARCHAR(64) NOT NULL,
Mobile VARCHAR(32) NULL,
Remark VARCHAR(255) NULL,
```

```
  ⑨  (Name, Birthday)
)
```

请将上述语句补充完整。

除了主键约束和 UNIQUE 约束，上述语句还使用了 NULL 和 NOT NULL 约束。

在这样众多的约束下，如果表 CorrMan 当前包含的数据如图 5-18 所示：

ID	Name	Birthday	Mobile	Remark
1	张三五	1991-01-03	139792382	NULL
2	刘俊生	1991-02-03	139792383	NULL
3	刘德华	1991-03-03	139792384	NULL
4	张学友	1991-04-03	139792385	NULL
5	小天王	1991-05-03	139792386	NULL
6	冯小刚	1991-06-03	139792387	NULL

图 5-18　表 CorrMan 中的数据示意图

则在多个 INSERT 语句中，只有语句 ⑩ 能够成功执行；在多个 UPDATE 语句中，只有语句______能够成功执行，并且能够修改一条记录的内容。

⑤ A．操作系统　B．关系数据库　C．数据文件　D．函数

⑥ A．表　B．域　C．数据库　D．字段

⑦ A．表　B．域　C．数据库　D．字段

⑧ A．NULL　B．UNIQUE KEY　C．PRIMARY KEY　D．UNIQUE

⑨ A．NULL　B．UNIQUE KEY　C．PRIMARY KEY　D．UNIQUE

⑩ A．INSERT INTO CorrMan (ID, Name, Birthday, Mobile)
　　VALUES (3, '张三丰' '2003-10-20', '133842343')

　B．INSERT INTO CorrMan (ID, Name, Birthday, Mobile)
　　VALUES (13, '张三丰', '2003-10-20', '133842343')

　C．INSERT INTO CorrMan
　　VALUES (13, '张三丰', '2003-10-20', '133842343')

　D．INSERT INTO CorrMan
　　VALUES (3, '张三丰', '2003-10-20', '133842343', '')

　A．UPDATE CorrMan SET ID = 2

　B．UPDATE CorrMan SET Name = '张二丰' WHERE ID = 2

　C．UPDATE CorrMan SET ID = 2 WHERE Name = '冯小刚'

　D．UPDATE CorrMan SET ID = 2 WHERE Name = '张二丰'

二、请阅读下面的文字，回答文字后面提出的问题，并将问题的答案填写在答题纸相应的地方。

【文字描述】 某学校使用了学生信息管理系统，其中对班级和学生的关系模型描述如下：学生（学号、姓名、性别、出生年月、民族、所属班级、宿舍）、班级（班级编号、班级名称、班长学号、班级人数、成立年月）关系的主要属性的含义和约束如图 5-19 所示，学生和班级关系分别如图 5-20 和图 5-21 所示。

属　性	含　义	约　束
学号	标识学生的编号	每个学生都必须存在一个唯一非空的标识，班级人数>0
班级编号	标识班级的编号	每个班级都必须存在一个唯一非空的标识，班级人数>0
所属班级	学生所属的班级的编号	每个学生都必须属于一个班级
班长学号	班级的班长的学号	班长也是学生
班级人数	所有属于相同班级的学生的总数	班级人数 >= 0，默认值为 0
姓名	学生的姓名	学生必须拥有非空姓名（允许同名）
班级名称	班级的名称	班级必须拥有非空名称（不允许同名）

图 5-19　属性的含义和约束

学　号	姓　名	性　别	出 生 年 月	民　族	所 属 班 级	宿　舍
1	张三五	男	1977-12	汉	1	27#317
2	刘俊生	男	1976-11	汉	1	27#317
3	刘德华	男	1978-12	汉	1	27#317
4	张学友	男	1979-10	汉	1	27#317
5	小天王	女	1980-03	蒙古	2	26#125
6	冯小刚	女	1973-05	汉	2	26#125

图 5-20　“学生”关系示意图

班 级 编 号	班 级 名 称	班 长 学 号	班 级 人 数	成 立 年 月
1	××学校物理 61 班	2	4	1991-01-03
2	××学校英语 72 班	6	2	1991-02-03

图 5-21　“班级”关系示意图

【问题 1】

根据上面的“文字描述”，可以使用 SQL 语句来定义如图 5-20 和图 5-21 描述的“学生”和“班级”关系。下面就是这样的 SQL 语句，请补充完整。

```
CREATE TABLE tblClass(
    ID INT _____,
    Name VARCHAR(16) _____,
    Monitor INT REFENENCES tblStudent,
    MemberNum INT ______,
    Birthday VARCHAR(16) NULL

)
```

```
CREATE TABLE tblStudent(
    ID INT ______,
    Name VARCHAR(16) ______,
    Sex VARCHAR(2) NULL,
    Birthday VARCHAR(16) NULL,
    Nation  VARCHAR(16) NULL,
    ClassID INT,
    Dormitory VARCHAR(64) NULL,
    ____(ClassID)______
)
```

【问题 2】

关系“班级”还包含一个用来统计班级拥有学生的数量的属性，即“班级人数”（MemberNum），这是一个统计类型的属性，即必须在向“学生”关系中插入数据或者修改其中的记录时，重新统计班级所拥有的人数，这就需要一个触发器来实现这个功能，而触发器需要执行的 SQL 语句（假设需要统计人数的班级的 ID 为 iid）如下所示，请补充完整。

```
UPDATE tblClass SET MemberNum=
(SELECT ______ FROM tblStudent WHERE ClassID=iid)
```

实际上，很容易理解，这是一个效率很低的方法，触发器中执行嵌套查询是不得已的做法。在这里涉及的问题中，完全可以使用另外一种方法，在默认值为 0 的情况下，向 tblStudent 中插入新的数据或者修改原来的数据对应的学生所属的班级时，通过触发器修改 MemberNum 的值。

另外，还可以使用分组查询的方法从 tblStudent 表中统计出所有班级包含的人数，并按照班级编号的顺序给出结果，下面就是这个 SQL 语句，请补充完整。

```
SELECT ClassID, ______ FROM tblStudent ______ ORDER BY ______
```

【问题 3】

使用问题 1 定义 SQL 语句创建的表中已经包含了如图 5-20 和图 5-21 所示的数据，执行下面的 SQL 语句可能还可以插入新数据或者修改其中的旧数据，但是由于约束的限制，导致某些 SQL 语句由于不满足约束的限制而无法成功执行，请指出哪些语句不能成功执行，并指出原因。

```
① UPDATE tblStudent SET ClassID=2 WHERE Name='刘俊生'
② INSERT INTO tblStudent(ID, Name, ClassID) VALUES(7, '小天王', 2)
③ INSERT INTO tblStudent(ID, Name, ClassID) VALUES(8, '小天王', 3)
```

【问题 4】

视图是常用的手段，可以在用问题 1 的两个 SQL 语句创建的表的基础之上创建一个视

图，下面就是创建视图的 SQL 语句：

```
CREATE VIEW vwStudent(ID, Name, Sex, Birthday, ClassName, Dormitory)
AS
SELECT tblStudent.ID, tblStudent.Name, Sex, tblStudent.Birthday, tblClass.
Name, Dormitory
FROM tblStudent, tblClass WHERE tblStudent.ClassID=tblClass.ID
```

在这样的视图上可以执行许多 SQL 语句，下面给出的 SQL 语句中，哪些不能在这个视图上执行，原因是什么？

```
① UPDATE vwStudent SET ClassID=2 WHERE Name='刘俊生'
② DELETE FROM vwStudent WHERE ID<3
③ SELECT COUNT(*) FROM vwStudent WHERE ID > 3
④ SELECT * FROM vwStudent WHERE ID=3
⑤ SELECT ClassName, COUNT(*) FROM vwStudent GROUP BY ClassName ORDER BY
ClassName
⑥ DELETE FROM vwStudent WHERE ClassName='abc'
⑦ UPDATE vwStudent Name='abcdef' WHERE ID=2
```

【问题 5】

下面是一段查询 tblStudent 表的 SQL 语句：

```
SELECT * FROM tblStudent
WHERE NOT EXISTS(SELECT * FROM tblClass WHERE Monitor=tblStudent.ID)
```

① 请用简短的文字描述这个 SQL 语句的功能。

② 对这个 SQL 语句的效率做简短分析，并给出一个能够实现类似功能而效率有所改善的 SQL 语句。

【问题 6】

使用 CREATE INDEX 语句可以为表中的字段创建索引，从而提高检索速度。

① 请分别写出为表 tblStudent 的 ID 字段和 Birthday 字段创建索引的 SQL 语句。

② 在①中创建的索引会改善下面两个查询语句的检索效率吗？

2-1　SELECT * FROM tblStudent WHERE ID > 3

2-2　SELECT * FROM tblStudent WHERE Birthday LIKE‘2001-%’

练习题答案

一、

① D

② B
③ B
④ B
⑤ B
⑥ A
⑦ D
⑧ C
⑨ D
⑩ B
　B

二、

【问题 1】

PRIMARY KEY
NOT NULL
DEFAULT 0
NOT NULL UNIQUE
FOREIGN KEY (ClassID) REFERENCES tblClass (ID)

【问题 2】

COUNT(*)
COUNT(*)
GROUP BY ClassID
ClassID

【问题 3】

① 可以
② 可以
③ 不可以，因为目前没有 ID 为 3 的班级，外键约束将导致无法执行

【问题 4】

① 不可以，视图中没有 ClassID 字段
② 不可以，删除操作将同时涉及到多个表
③ 可以，视图上可以执行查询语句
④ 可以，同上
⑤ 可以，视图上可以执行分组查询
⑥ 不可以，删除操作将同时涉及到多个表
⑦ 可以，这种操作仅仅涉及到一个表，如果符合值约束的条件，则可以执行

【问题 5】

① 列出所有不是班级班长的学生的信息

② 因为使用了 NOT EXISTS 语句，所以对于 tblStudent 表中的每一个记录，都需要执行一次子查询操作，这会导致效率很低，下面给出改进这个 SQL 语句效率的方法：

```
SELECT * FROM tblStudent
WHERE ID NOT IN(SELECT DISTINCT(Monitor) FROM tblClass)
```

【问题 6】

① 下面就是创建索引的 SQL 语句：

```
CREATE INDEX IndexStudentID ON tblStudent(ID ASC)

CREATE INDEX IndexStudentBirthday ON tblStudent(Birthday ASC)
```

②

2-1　能

2-2　不能

第 6 章　网络环境下的数据库

本章提示

随着网络技术的迅速发展和应用，人们可以通过网络访问异地资源，实现在地域分散情况下的数据传输，达到数据共享的目的。因此，网络环境下的数据库成为现今数据库研究和应用的主要方面之一。本章根据考试大纲的要求，给出了网络环境下分布式数据库的体系结构设计、数据库系统的设计与实施的策略与技术、面向 Web 的 DBMS 技术这 3 个主要知识点的体系结构图，同时对其中的重点和难点给出详略恰当的提炼。在详细的典型例题分析之后，还给出了适量的实战练习题，以加深对这些知识的理解。本章共分两节，如图 6-1 所示的是本章的知识框图。

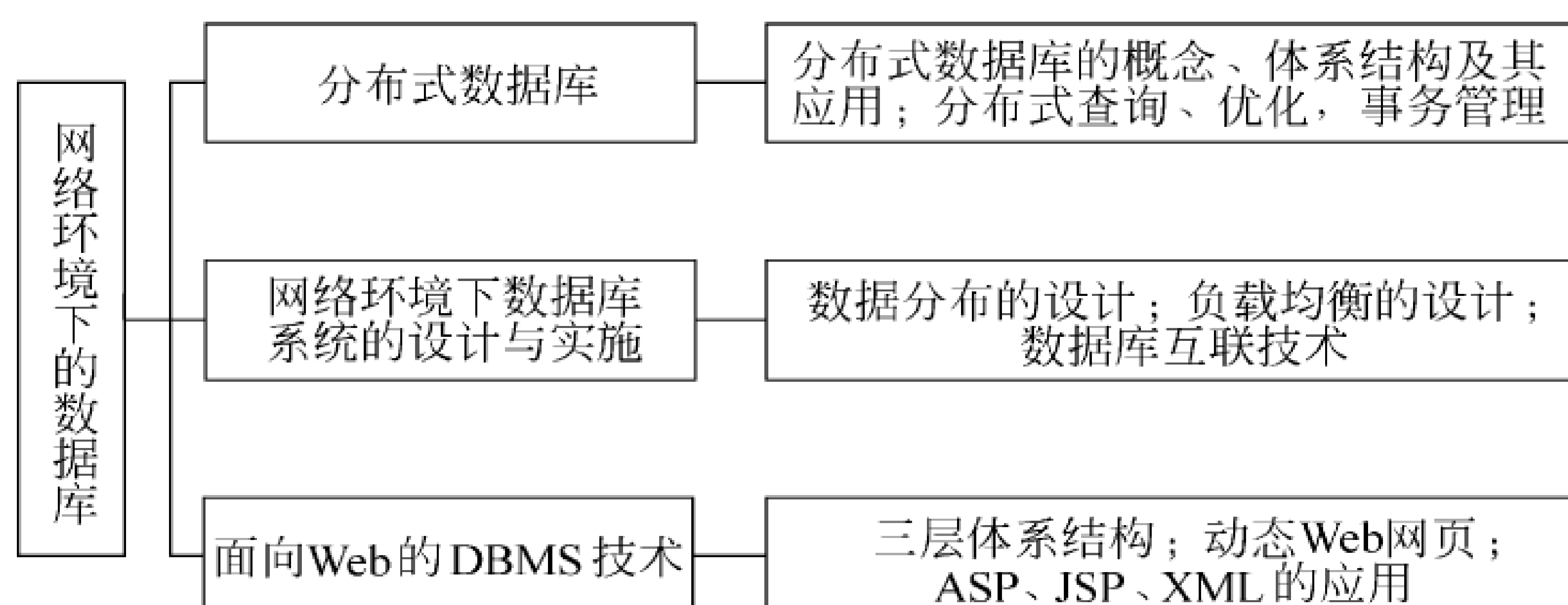


图 6-1　网络环境下数据库知识框图

从图 6-1 可以大体了解网络环境下与数据库系统相关的知识体系，按照大纲给出的要求，本章对分布式数据库系统做了更为详细、深入的阐述，同时也对网络环境下的数据库设计、实施和实际应用中比较成熟的面向 Web 的 DBMS 技术做了一定的介绍。

6.1　分布式数据库

分布式数据库由一些松耦合的节点组成，每个节点都可以参与事务的执行，这些事务所访问的数据可以位于一个节点也可以位于几个节点上。数据的分布很大程度上提高了整个系统的数据可用性、每个节点的数据可调控性，但同时也给事务处理和查询处理带来很多额外开销。

作为网络环境下的可选数据库系统之一，我们必须明晰分布式数据库系统的体系结构、数据分布策略，熟悉在实际应用中如何对分布式数据库进行设计、管理、数据查询、优化等工作。同时还要与集中式数据库进行横向对比，以便在网络环境下选择正确的数据库系统。

本小节系统、全面地介绍了分布式数据库，在较为详细地给出整个分布式数据库系统的知识框架图（如图 6-2 所示）后，还在每一子小节给出具体的知识框架图，以期帮助读者能够全面直观地了解本节，参照大纲给出的具体要求，做到知识体系结构的深度和广度的均衡协调。

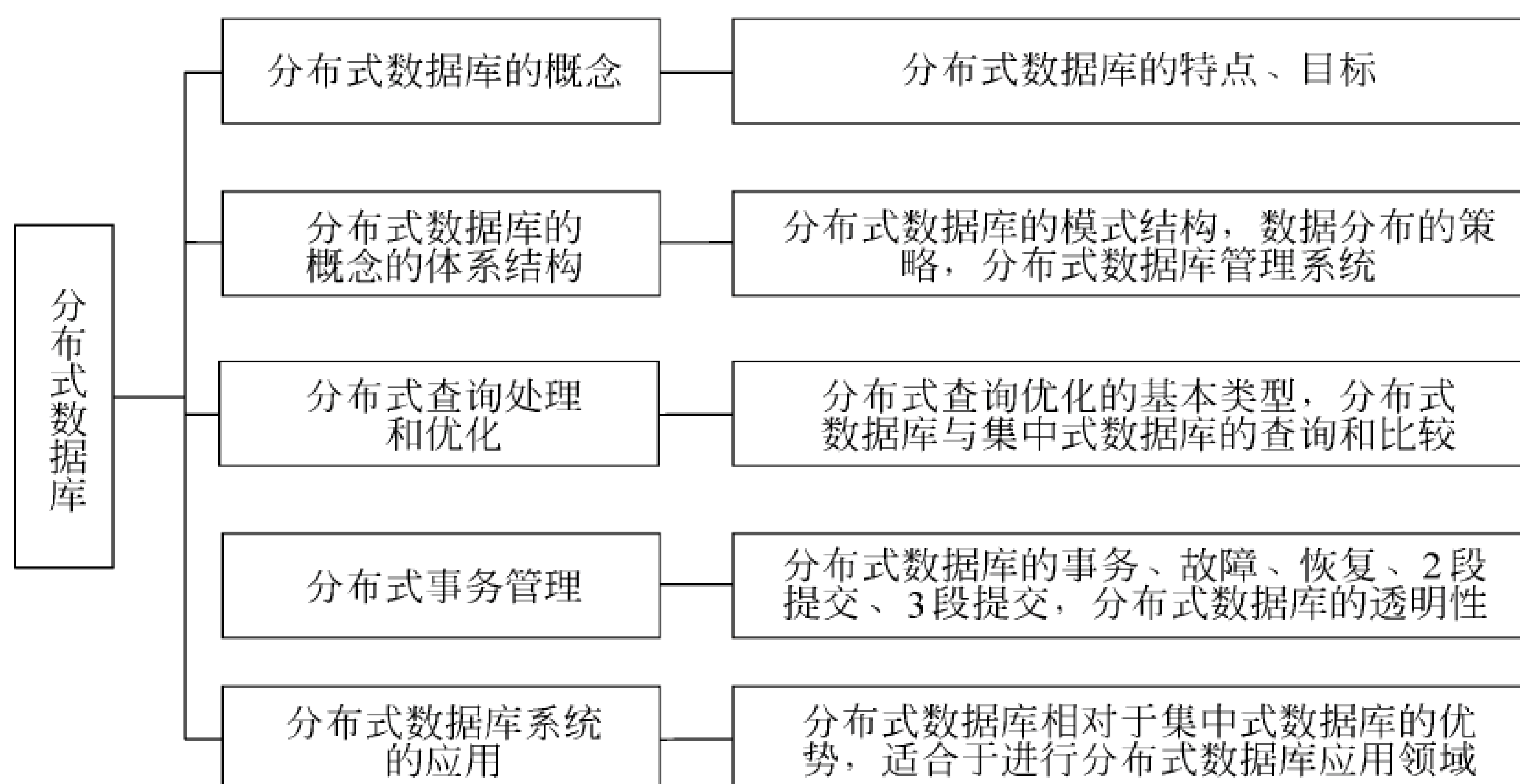


图 6-2　分布式数据库知识框架图

6.1.1　分布式数据库的概念

分布式数据库系统是数据库系统和计算机网络相结合的产物。一方面，由于计算机功能增强，成本下降，几乎每个办公室、实验室，个人用户都可以拥有自己的计算机，从而增加了数据分散处理的需求。另一方面，由于通信技术的迅速发展，出现了各种计算机网络，从而降低了数据传输费用，而计算机局部网络的广泛应用，则为分布式数据库系统的出现提供了实现的可能性。

1．知识点提炼

（1）分布式数据库的定义

分布式数据库系统（Distributed Database System，DDBS）是针对面向地理上分散，而管理上又需要不同程度集中的需求而提出的一种数据管理信息系统。满足下面条件的数据库系统被称为完全分布式数据库系统。

① 分布性：即数据存储在多个不同的节点上。

② 逻辑相关性：即数据库系统内的数据在逻辑上具有相互关联的特性。

③ 场地透明性：即使用分布式数据库中的数据时不需指明数据所在的位置。

④ 场地自治性：即每一个单独的节点能够执行局部的应用请求。

（2）分布式数据库的特点

分布式数据库系统，是传统集中式数据库系统的发展，因此它具有集中式数据库系统的特点，同时，由于它的分布性而又使这些特点具有新的含义。下面给出了分布式数据库的几个主要的特点。

① 数据的集中控制性

由于分布式数据库系统是在传统数据库系统的基础上发展起来的，因此，它也具有集中控制的特性。在分布式数据库中可以认为存在全局数据库管理员和局部数据库管理员，这是一种分层控制结构，一般来说，全局数据库管理员负责管理所有数据库，而局部数据库管理员只负责各自节点的局部数据库，但是在有些情况下，局部数据库管理员可以有更高的自主性，甚至完成节点间的协调工作，从而不再需要全局数据库管理员。

② 数据独立性

数据独立性也是集中式数据库和文件系统相比所具有的一大特征，独立性是指数据的组成对应用程序来说是透明的。应用程序只需要考虑数据的逻辑结构，而不用考虑数据的物理存放，因而数据在物理组织上的改变不会影响应用程序。

③ 数据冗余可控性

分布式数据库中的数据一般存储在经常使用的场地上，但两个或两个以上的场地应用对同一数据有存取要求也是时常发生的，而且当传输代价高于存储代价时，可以将同一数据存储在两个（甚至更多）场地上，以节省传输的开销。另外，数据有多个副本，也可以提高系统的可用性，即当系统中某个节点发生故障时，因为数据有其他副本在非故障场地上，对其他所有场地来说，数据仍然是可用的，从而保证数据的完备性。因为这种冗余度是在系统控制之下的，所以给系统造成的不利影响是可控制的。

最后，由于可用副本的存在也相应地提高了场地自治性。

④ 场地自治性

在分布式数据库系统中，多个场地的局部数据库在逻辑上集成为一个整体，这个整体被称为全局数据库，并为分布式数据库系统的所有用户使用，这种应用称为分布式数据库的全局应用，其用户为全局用户；同时，分布式数据库系统还允许用户只使用本地的局部数据库，这种应用为局部应用，其用户为局部用户，甚至局部用户所使用的数据可以不参与到全局数据库中去。这种局部应用独立于全局应用的特性就是局部数据库的自治性。

由于自治性，对每个场地来说就有两种数据，一种是参与全局数据库的局部数据，而另一种则是不参与全局数据库的数据。

⑤ 存取的有效性

分布式数据库系统中的全局查询被分解成等效的子查询，即全局查询的执行计划分解成多个子查询执行计划加以执行，它是根据系统的全局优化策略产生的，而子查询计划又是在各场地上分布执行的。因而，分布式数据库系统中查询优化有两个级别：全局优化和局部优化。

全局优化主要决定在多个副本中选取合适的场地副本，使得场地间的数据量传输次数最少，从而使系统通信开销少。而局部优化和传统的集中式数据库中的优化是一致的。

（3）分布式数据库的目标

分布式数据库实施的目标是将系统中分布在不同节点上的数据库系统实现数据的共

享，达到数据利用率高、存取快、可用性高，用户操作这些数据如同在本地数据库使用。

2．难点分析

分布式数据库与其他数据库系统或应用系统的区分

上节提到只有满足了分布性、逻辑相关性、场地透明性和场地自治性的数据库系统才能成为真正意义上的分布式数据库。为什么提出这 4 个基本条件呢？这是因为分布式数据库系统的分布性可以用于区分单一的集中式数据库与分布式数据库；根据逻辑相关性，则可以将分布式数据库与一组局部数据库或存储在计算机网络中不同节点的文件系统区分开来；根据场地透明性和场地自治性则可以和多机处理系统或并行系统区分开来。

6.1.2　分布式数据库的体系结构

分布式数据库的模式结构是目前国内外尚在讨论的问题，还没有统一的标准，这是因为对分布式数据库系统数据独立性要求的程度不同，其抽象层次也不同。我国在多年的研究与开发分布式数据库的基础上制定了《分布式数据库系统标准》，此后讨论的任何有关分布式数据库的知识要点都是基于此项标准。

1．知识点提炼

（1）分布式数据库的模式结构

我国制定的《分布式数据库系统标准》中，曾提出把分布式数据库抽象为 4 层的模式结构，如图 6-3 所示。这一模式结构得到了国内外一定程度的支持和认同。

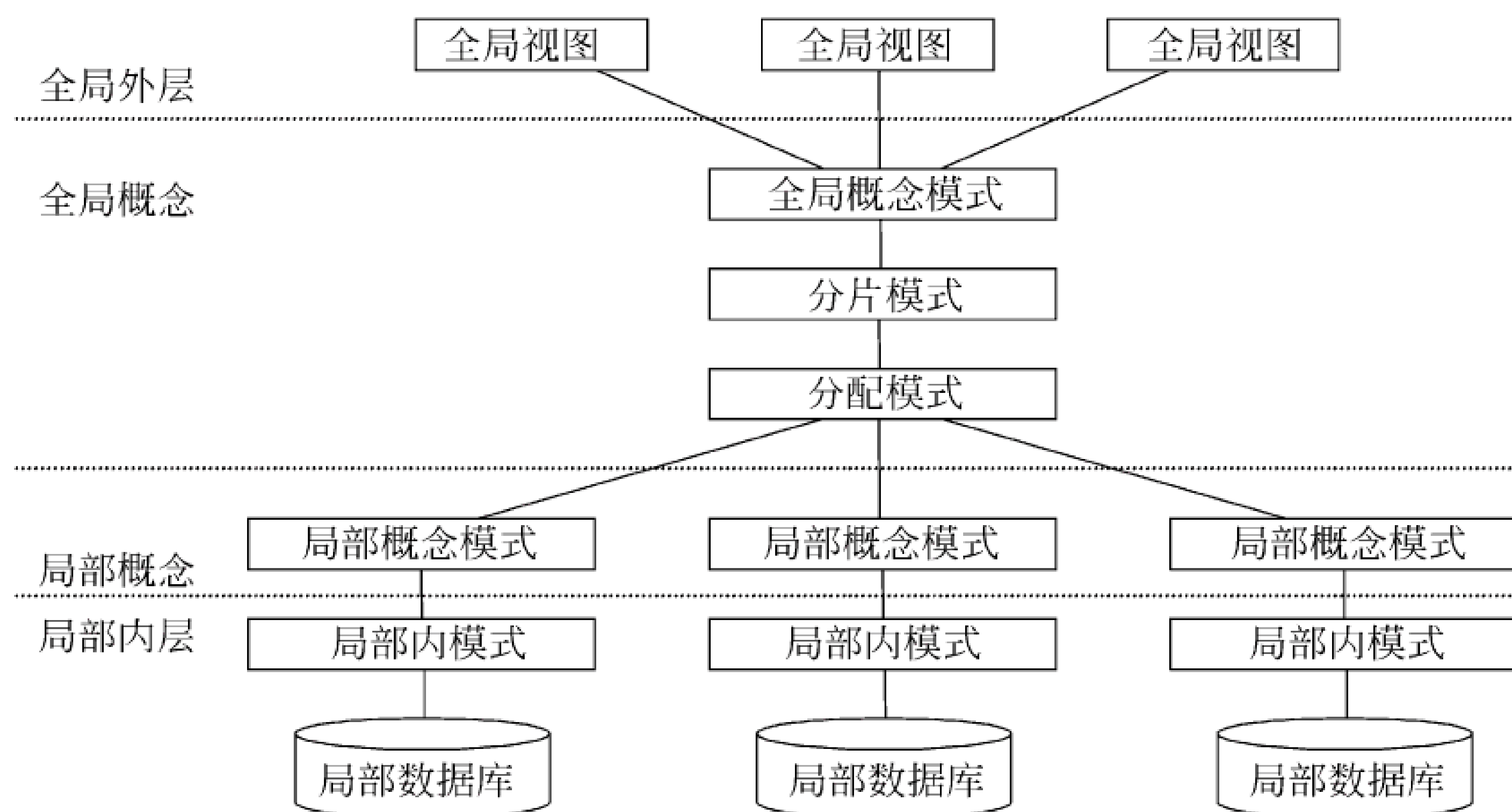


图 6-3　分布式数据库模式结构框图

这种 4 层模式划分为：全局视图、全局概念层、局部概念层和局部内层。在各层间还有相应的层间映射。4 层模式的划分不仅适用于完全透明的分布式数据库系统，而且也适合各种透明性要求的分布式数据库系统。无论是对同构型分布式数据库系统，还是对异构型分布式数据库系统都能适用。

① 全局外层

分布式数据库是一组分布的局部物理数据库的逻辑集合。分布式数据库的全局视图如同集中式数据库一样，由多个用户视图组成。只不过此处的用户视图是针对分布式数据库特定的全局用户而言，是对分布式数据库的最高层的抽象。

分布式数据库与集中式数据库的视图有同样的概念，不同的是，它不是从某一个具体场地上的局部数据库中抽取，而是从一个虚拟的由各局部数据库逻辑集合中抽取，对全局用户而言，不论它在分布式数据库系统中的哪一个节点上访问系统中的数据，都可以认为所有的数据库都在本场地，而且它只关心它们自己所使用的那部分数据。

如果是完全透明的关系模型的分布式数据库结构，则视图就和集中式数据库的视图一样，其定义方式也基本相同，因此全局用户在使用视图时，就不必关心数据的分片和具体的物理细节。若为非完全透明的分布式数据库，则在视图定义中，根据透明性支持的程度，需要给出一定的数据细节、物理存取的细节等。

② 全局概念层

全局概念层是分布式数据库的整体抽象，包含了系统中全部数据的特性和逻辑结构。就像集中式数据库中的概念视图一样，是对数据库的整体的描述，但在分布式数据库的 4 层抽象的结构中，全局概念层比集中式的概念层有更多的描述。

从分布透明特性来说，分布式数据库的全局概念层应具有 3 种模式描述信息，如下。

- 全局概念模式：描述分布式数据库全局数据的逻辑结构，是分布式数据库的全局概念视图。与集中式数据库的概念视图的定义相似，全局概念模式应包含模式名、属性名以及每种属性的数据类型的定义和长度。
- 分片模式：描述全局数据逻辑划分的视图，它是全局数据的逻辑结构根据某种条件的划分，每一个逻辑划分即是一个片段或称为分片。
- 分配模式：描述局部逻辑的局部物理结构，是划分后的片段（或分片）的物理分配视图。它与集中式数据库物理存储结构的概念不同，是全局概念层的内容。

③ 局部概念层

局部概念层由局部概念模式描述，一般情况下，它是全局概念模式的子集，全局概念模式经逻辑划分后被分配在各局部场地上。

在分布式数据库局部场地上，每个全局关系有该全局关系的若干个（可允许是全部）逻辑片段的物理片段集合，该集合是一个全局关系在某个局部场地上的物理映像，全部的物理映像组成局部概念模式。如果两个场地上的所有物理映像都相同，则其中一个场地上的数据必然是另一个场地的副本，两个场地的局部概念模式亦相同。

如果分布式数据库只支持全局应用，则局部概念模式可理解为局部数据库的概念模式和外模式，在此情况下，外模式和概念模式是相同的；如果分布式数据库还支持局部用户，而局部用户定义的局部数据不参与分布式数据库的全局数据，则局部概念层还应划分为局部外模式和局部模式，并且由局部 DBA 描述，这些将不属于全局概念模式。这时值得注

意的是，全局数据和局部数据的管理分别由全局 DBA 和局部 DBA 管理。因此，全局用户是否可以使用全局数据则由全局 DBA 授权，局部 DBA 无权授予全局用户各种权限。

当全局数据模型与局部数据模型不同时，则物理映像与各局部数据库的数据模型之间还必须有数据模型的转换。即使是数据模型相同，也可能存在数据类型和格式的各种转换。也就是说，各局部数据库是多种数据模型构成的数据库时，若组成分布式数据库，则需要一个统一的全局描述，即数据模型的同种化的集成，而对不同规格化的统一，则称之为一体化。这就是分布式数据库中的全局概念层到局部概念层的映射模式的描述。

④ 局部内层

局部内层是分布式数据库中关于物理数据库的描述，相当于集中式数据库的内层。其描述的内容和方法与之大致相同。

（2）数据分布的策略

数据分布的策略方面主要的工作是围绕着两方面进行的：“高效的数据划分问题”和“数据放置问题”。第一个方面是关于如何把数据划分开，使得使用率最高的数据能够被放置在性能最好的场地上。第二个方面是关于如何把已划分好的数据合理地放置在网络上以获得最好的执行效率，减少网络传输的数据量。数据的划分和放置是数据分布问题的两个方面，只解决其中任何一个都不能说是已经解决了数据分布问题。数据分布是分布式数据库的特征，解决数据分布的策略一般有以下几种。

① 集中式：所有全局数据片段都安排一个节点上。

这种分布策略是把系统数据存放在一个节点上，对数据的控制和管理都比较容易，数据的一致性和完整性能够得到保证。但是由于数据的检索和修改都必须通过这个节点，使得这个节点的负担过重，容易出现瓶颈。另外，系统对这个节点的依赖性也过多，一旦这个节点出现故障，将使整个系统崩溃，系统的可靠性就相对较差，为了提高系统的可靠性，该节点的设备就必须提高。

② 分割式：所有全局数据有且只有一份，它们被分割成若干个逻辑片段，每个逻辑片段被分别指派在特定的节点上，可以说对全局数据进行了划分。

这种分布策略充分利用各个站点上的存储设备，数据的存储量大。在存放数据的各个节点可自治地检索和修改数据，发挥系统的并发操作能力。同时，因为数据是分布在多个节点上的，所以当某部分节点出现故障时，系统仍可运行，提高了系统的可靠性。对于全局查询和修改，所需的时间会比集中式长些，因为数据不在同一场地上，需要进行网络通信。

③ 复制式：全局数据有多个副本，每个站点上都有一个完整的数据副本。

采用这种策略的系统可靠性较高，响应速度快。数据库的恢复也较容易，可从任意的场地得到数据的副本。但是要保持各个站点上数据的同步修改，将要付出昂贵的代价。另外，整个系统的数据冗余很大，系统的数据容量也只是一个节点上数据库的容量。

④ 混合式：全部数据被分为若干个数据子集，每个子集被放在不同的节点上，但任

何一个节点都没有保存全部的数据，根据数据的重要性决定各个数据子集副本的数量。

这种分布策略，兼顾了分割式和复制式的做法，也获得了二者的优点，它灵活性好，能提高系统的效率，但同时也包括了二者的复杂性。

（3）数据分片

在分布式数据库中，数据存放的单位是数据的逻辑片段。对关系型数据库来说，一个数据的逻辑片段是关系的一部分。数据分片有 3 种基本方法，它们是通过关系代数的基本运算来实现的。

① 水平分片：按特定条件把全局关系的所有元组，分划成若干个互不相交的子集，每一子集为全局关系的一个逻辑片段。它们通过对全局关系施加选择运算得到，并可通过对这些片段执行合并操作来恢复该全局关系。

② 垂直分片：把全局关系的属性分成若干子集，对全局关系进行投影运算得到这些子集。要求全局关系的每一属性至少映射到一个垂直片段中，且每一个垂直片段都包含该全局关系的关键字。这样，通过对这些片段执行连接操作就可以恢复该全局关系。

③ 水平和垂直结合的分片：以上两种方法的混合。可以先水平分片再垂直分片，或先垂直分片再水平分片。

不论采用哪一种方法进行数据分片，都要要遵守如下规则。

① 完备性条件：必须把全局关系的所有数据映射到各个片段中，绝不允许有属于全局关系却不属于任何一个片段的数据存在。

② 可重构条件：必须保证能够由同一个全局关系的各个片段来重新构造该全局关系。对于水平分片可用并操作重构全局关系；对于垂直分片可用连接操作重构全局关系。

③ 不相交条件：要求一个全局关系被分割后所得的各数据片段互不重叠或只包含关键字重叠。

（4）数据分布透明性

在分布式数据库中分布独立性也称为分布透明性。下面来看看分布透明性的各种级别。

① 分片透明性

分片透明性是分布透明性中的最高层，在 4 层分布式数据库模式结构中，分片透明性位于全局概念模式与分片模式之间。当分布式数据库具有分片透明性时，用户编写的应用程序只对全局关系进行操作，而不必考虑数据的逻辑分片，当分片模式改变时，只要改变全局概念模式到分片模式之间的映像即可，从而不会影响应用程序，实现了数据分片透明性。

② 分配透明性

分配透明性分布透明性的中间层，在 4 层的分布式数据库模式结构中，位于分片模式与分配模式之间。实际上，分配透明性包含了两种情形：一种是各片段被复制的情况，即每一片段是否被复制、复制了几个副本；另一种是片段及其各副本的场地的位置分配情况。前者也称复制透明性或数据冗余透明性。当分布式数据库具有分配透明性时，用户编写的

应用程序要了解全局数据的数据分片情况，但不必了解各逻辑片段的复制副本情况，也不必关心各片段及其副本的站点位置分配情况。当片段及其副本的存储站点改变时，只要改变从分片模式到分配模式之间的映像即可，从而不会影响用户程序，实现了数据片段的位置透明性。

③ 局部数据模型透明性

局部数据模型透明性也称局部映像透明性，即与各场地上数据库的数据模型无关，是分布透明性的最低层，在 4 层分布式数据库模式结构图中，处于分配模式与局部概念模式之间。当分布式数据库只具有局部数据模型透明性时，用户编写应用程序不但要了解全局数据的逻辑分片情况，还要了解各逻辑片段的副本复制情况，以及各片段和它们副本的站点位置分配情况，但不必了解各站点上数据库的数据模型。全局数据模型与每个节点上的局部数据库的数据模型的转换是由分配模式与局部概念模式之间的映像实现的。当某个节点上数据库的数据模型改变时，只要改变分配模式到该站点局部概念模式之间的映像即可，应用程序不受影响，从而实现了局部数据模型透明性。显然，在同构分布式数据库系统中，其各站点上的数据模型相同，且有可能全局数据库的数据模型就采用局部数据库的数据模型，此时，就大大减少了这种映像的复杂性。

（5）分布式数据库系统

关系数据库的最早设计者提出了完全的分布式数据库管理系统应遵循的 12 条规则，这 12 条规则已被广泛接受，并作为分布式数据库系统的标准定义。它们是：

① 场地自治性

② 非集中式管理

③ 高可用性

④ 位置独立性

⑤ 数据分割独立性

⑥ 数据复制独立性

⑦ 分布式查询

⑧ 分布式事务管理

⑨ 硬件独立性

⑩ 操作系统独立性

网络独立性

数据库管理系统独立性

如果一个分布式数据库管理系统能够满足上面的 12 条准则，那么就可称这个分布式管理系统为完全的分布式管理系统

2．难点分析

（1）如何利用数据分布透明性来配置和开发分布式数据库实际应用

如果一个分布式数据库系统提供分片透明性，它一定也提供了分配透明性和局部数据

模型透明性，所以也称为完全分布透明性，是分布透明性的最高级别。此时，对用户和用户程序而言，他们所面对的分布式数据库系统，如同集中式数据库一样，不必考虑数据的分片细节、不必考虑各片段的副本情况、不必考虑片段及副本的分配细节，也无须考虑各站点上数据库是什么数据模型等。

如果一个分布式数据库系统提供分配透明性，而没有提供分片透明性，它一定也提供局部数据模型透明性，所以也称为中级分布透明性，此时，对用户和应用程序而言，他们必须知道分布式数据库全局数据的逻辑分片情况，在程序中必须指出所需要访问的逻辑片段名。但不必关心逻辑片段是否被复制以及它们被分配在哪些站点上，也不必考虑站点的数据模型。

如果一个分布式数据库系统只提供局部数据模型透明性而不提供分片透明性，也不提供分配透明性，则被称为低级分布透明性。此时，对用户和应用程序而言，他们不但必须知道分布式数据库全局数据的逻辑分片情况，还必须知道各片段是否有副本、有多少副本、各片段及其副本被分配在哪些站点上。即在程序中不但要指定要访问的数据逻辑片段名，因此，要指定它们所在的节点名。但不必考虑站点上的数据模型。

如果一个分布式数据库系统，连局部数据模型透明性也不提供，即将异构数据模型转换也交给用户和用户程序自己处理。这种分布式数据库系统称为无分布透明性。

由此可见，一个分布式数据库系统可能提供的分布透明性的层次越高，用户编写应用程序越容易。因此，一个分布式数据库系统可能提供的分布透明性的程度，也是衡量分布式数据库管理系统是否完善的标准之一。

（2）我国制定的《分布式数据库系统标准》如何满足分布式数据库的基本特点

① 全局数据库与局部数据库分离。全局数据库是虚拟的，全局 DBA 的视图由全局概念层定义，完全独立于各个场地的局部数据库；局部概念层和局部内层可看作是局部数据库，它是全局数据库的内层。这样，不论是同构或异构型的分布式数据库，其全局数据库到局部数据库都是由映射模式解释的，所不同的是，同构型分布式数据库比异构型分布式数据库在映射模式上的复杂性较低。而对于全局用户来说，他们所关心的只是外层所定义的视图，他们只需使用全局数据所提供的语言去操纵分布式数据库，无须考虑各种模型转换、语言的转换、场地分配等细节。全局数据库与局部数据库的分开描述，不仅体现了本章中关于分布式数据库的定义，同时也体现了它具有模式转换的透明性。

② 数据库的数据独立性。4 层结构中的全局外层是数据库的用户视图，可有多个。全局概念层和局部概念层是分布式数据库的全局整体逻辑数据和局部整体逻辑数据的抽象，由于分布式数据库的分布特性决定了全局整体逻辑数据的抽象只有一个，而局部数据的逻辑抽象则是每个局部数据库各有一个，当然也允许其中的某些逻辑抽象完全相同，这样，分布式数据库就具有了集中式数据库那样的数据独立性——逻辑数据独立性和物理数据独立性。

③ 透明性。在全局概念层中，把数据的分片概念和数据的分配概念分别定义，从而

把分布透明中的分片透明和分配透明相分离。所谓分片透明即用户完全只对全局关系操作，而不管关系如何在逻辑上划分成片段关系，在全局概念层把分片透明看作是最高程度的透明性。所谓分配透明，是较低级的透明，要求用户在片段上操作，不是在全局关系上的操作，但不必考虑片段的存放位置，对用户而言，在完全透明的情况下，系统支持由分片定义而选中所需的片段进行操作，并由系统选择出适当的场地执行。从而实现了对用户的分布完全透明。这种分离对分布式数据库设计是十分有利的，可在逻辑设计阶段考虑分片的划分要求，而在实现时才考虑数据分配问题。

④ 数据冗余控制。冗余只在分配时才涉及，并且分布式系统提供了重复副本透明性。分布式系统还可提供比场地透明更低一级的透明性管理，即用户只要指定某个副本，系统对其他副本完成相应的操作，从而保证所有副本的完整性和可用性。

6.1.3　分布式查询处理和优化

1．知识点提炼

（1）分布式数据库系统中查询处理的特点

分布式数据库中查询处理问题的基础是集中式关系数据库系统中的概念及策略。分布式查询处理从讨论分布式查询的特点入手，并假设分布式数据库管理系统提供完全透明性。

与在集中式数据库环境中的查询相比较，分布式数据库环境中的查询要增加对以下两个方面的考虑。

① 数据和信息均要通过通信线路进行传输，存在延迟的问题，从而会减慢整个查询的执行过程。

② 网络中多处理器的存在提供了并行数据处理和传输的机会，应充分利用以加快查询的速度。

（2）分布式数据库系统中查询的基本类型

查询优化的基本类型通常包括两类：针对查询执行代价的优化和针对查询响应时间的优化。执行代价是指查询所需要的系统资源；查询响应时间是指查询开始提交到获得第一个结果之间的时间。

2．难点分析

分布式数据库查询执行代价和响应时间的区别

一般情况下，查询响应时间对一个组织机构而言，往往就代表着执行代价。例如对于一个商业组织机构，因为响应时间的延误而失掉了销售额或其他机会，就意味着商业损失，但是，出于教学和编程方面的某些原因，这里需要将查询执行代价和响应时间加以区分。

针对查询执行代价进行优化的目标是，使查询执行所使用的系统资源的总和尽量地少，从而降低系统开销，整个系统的开销可以从各单个系统资源的开销表达式中推出。针对查询响应时间优化的目标是尽量减少查询的响应时间，而不计较系统资源的耗费。可以形象地说，执行代价优化的目标是“最便宜”，而响应时间优化的目标是“最快”。

6.1.4　分布式事务管理

1．知识点提炼

（1）分布式事务

在分布式数据库系统中，分布式事务表明一个要求访问多个站点上数据库中数据的事务，但不关心存放数据的具体地点。分布式数据库管理系统的事务优化器实现把一个分布式事务转变为若干个与相应站点有关的操作序列组成，这些操作序列也称为“子事务”。所以，在分布式数据库系统中，可以把一个分布式事务看成是由若干个不同站点上的子事务组成的。

（2）分布式数据库故障

在分布式数据库系统中，一般地，把网络上各节点可能出现的故障称为节点故障，它们包括集中式系统中可能发生的故障，而把各节点之间通信出现的故障称为通信故障。

通信故障可分为报文故障和网络分割故障。而报文故障又可分为报文错、报文失序、报文丢失和长时间的延迟。对报文错和报文失序现今网络都可检测和处理，所以通信故障主要是报文丢失、报文延迟和网络分割。下面对每一种故障一一进行解释。

① 介质故障：存放数据的介质发生的故障，如磁带、磁盘的损坏等。

② 系统故障：CPU 错、死循环、缓冲区满、系统崩溃等。

③ 事务故障：计算溢出、完整性被破坏、操作员干预、输入输出错等。

④ 网络分割故障：系统中一部分的节点和另外一部分节点完全失去了联系，两组节点无法通信。

⑤ 报文故障：收到的报文格式或数据错误、报文先后次序不正确、丢失了部分报文和长时间收不到报文。

综上所述，故障的分类如图 6-4 所示。

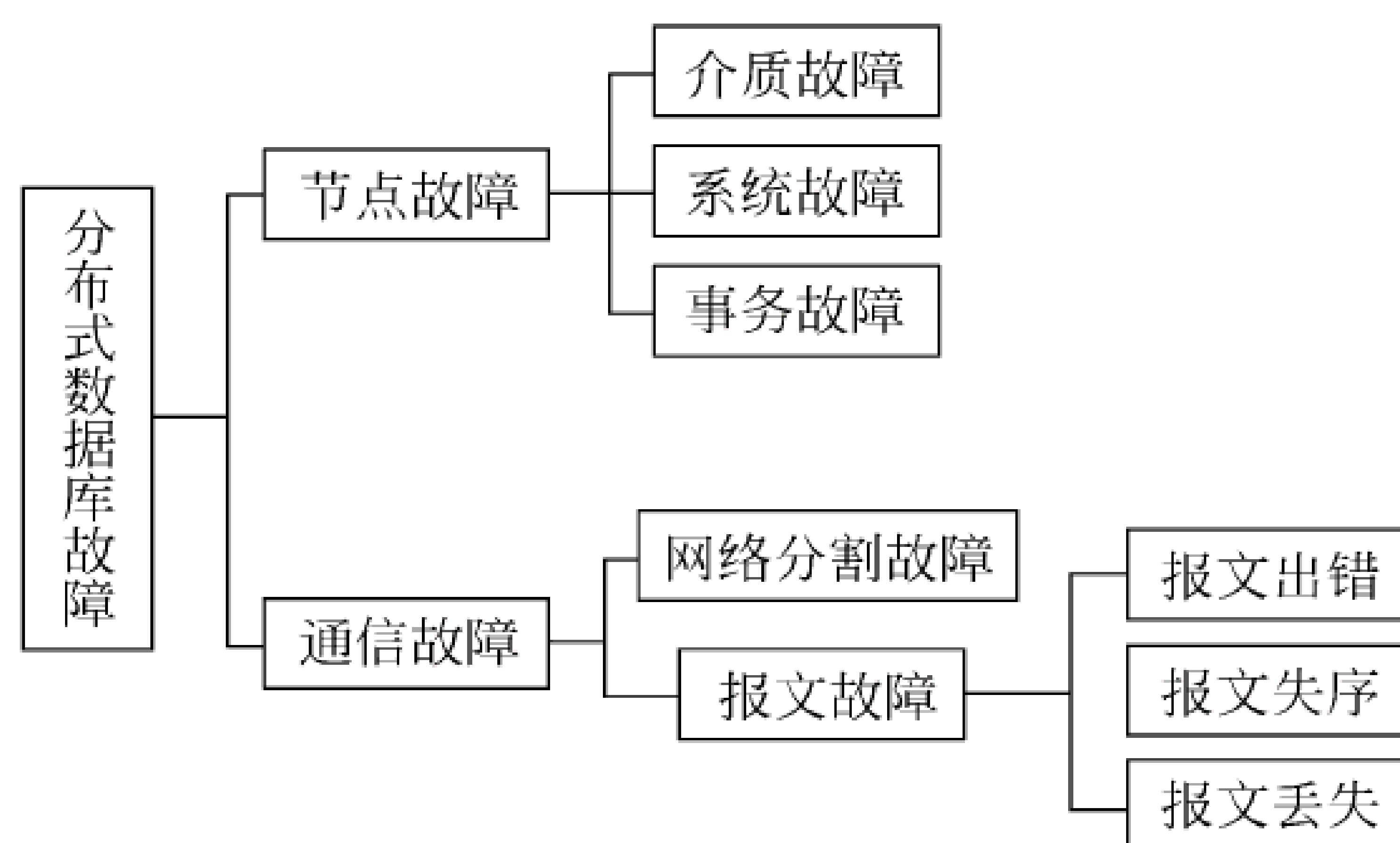


图 6-4　分布式数据库故障类别

（3）分布式数据库故障恢复的原则

在分布式数据库系统中，故障的类型很多。当发生事务故障时，保证事务原子性的措施称为事务故障恢复，简称为事务恢复。事务本身的故障和系统的故障是造成数据库完整

性和一致性被破坏的主要原因。当发生事务故障时，保证事务原子性的措施称为事务故障恢复，简称为事务恢复。事务恢复主要是依靠日志来实现的。恢复应遵循的原则如下。

① 孤立和逐步退出事务的原则

对于不影响其他事务的可排除性局部故障，例如事务操作的删除、超时、违反完整性规则、资源限制、死锁等，应令某个事务孤立地和逐步地退出，将其所做过的修改复原，即执行 UNDO。

② 成功结束事务原则

成功结束事务所做过的修改应超越各种故障，当故障发生时，应该重新执行（REDO）事务的所有操作。

③ 夭折事务的原则

若发生了非局部性的不可排除的故障，例如系统崩溃，则撤销全部事务，恢复到初态。这有两种做法，一种是利用数据库的备份实现；另一种是按反向顺序操作，复原其启动以来所做过的一切修改。

从集中式事务恢复可以了解事务恢复的一般过程，对分布式事务来说，由于处于网络环境下，其恢复处理远比集中式事务恢复要复杂得多。在分布式事务恢复中，本地事务的恢复和集中式事务的恢复相同，由本地事务管理器（Local Transaction Management，LTM）具体执行；而整个分布式事务的恢复由分布式事务管理器（Distribute Transaction Management，DTM）与 LTM 协同完成。

2．难点分析

（1）分布式事务与集中式数据库系统事务的对比

分布式事务和集中式数据库中的事务一样具有下面的特性。

① 原子性：事务的操作要么全部执行，要么全部不执行。当事务非正常终止时，其中间结果将被取消。事务的原子性保证数据库的状态总是从一个一致的状态变化到另一个一致的状态，而不会出现不一致的中间状态。

② 可串行性或一致性：并发执行的几个事务，其操作的结果应与以某种顺序串行执行这几个事务所得出的结果相同，因此称为可串行性。这种可串行化的并行调度是由数据库系统的并发控制机制来完成的，以保证并发事务执行时的数据库状态的一致。所以这种性质也称为事务的一致性。

③ 隔离性：一个没执行完的事务不能在其提交之前把自己的中间结果提供给其他的事务使用。因为未提交事务的结果不是最终结果，它有可能在以后的执行中被迫取消，如果其他的事务用到了它的中间结果，那么该事务也要夭折。

④ 持久性：当一个事务正常结束后，即提交后，其操作的结果将永久化，提交后发生的故障不会影响提交结果。即使发生了故障，系统应能够保证可以把事务的操作结果恢复过来。

人们常把事务的原子性（Atomicity）、可串行性（Serializability）或一致性（Consistency）、

隔离性（Isolation）和持久性（Durability）简称为事务的四性，即 ASID 或 ACID。

由于分布式数据库系统的分布特性，分布式事务的四性更带有分布执行时的特性。例如，在分布式数据库系统中，为了保证事务的原子性，组成这个分布式事务的各个子事务，要么全部都提交（成功结束），要么全都撤销（不成功结束），这就需要对各子事务进行协调和控制。此外，在分布式事务中，除了需要考虑访问数据互斥的存取操作序列外，还必须考虑大量的数据传送、通信原语和控制报文等，这些都是分布式事务所特有的性质。因此分布式事务与集中式数据库中的事务相比，在下面几个特性上有所区别。

① 执行特性：由于分布式事务执行时被分解成多个子事务执行，而各子事务间的操作需要进行协调，因此每一个分布式事务必须创建一个控制进程（亦称协调进程），以协调各子事务的操作，协调数据及控制报文的收发，决定事务的提交与夭折。而集中式事务的执行由并行调度算法调度，不必产生一个控制进程，也不必分解为子事务。

② 操作特性：在分布式事务中，除了应用对数据的存取操作序列之外，还必须加入大量的通信原语，负责协调进程和代理进程（负责完成子事务）之间的数据传送，以及代理进程之间的数据传送。此外，为了协调子事务的执行，还要加入大量的控制原语。因此，分布式事务比集中式事务的组成要复杂，而且执行的方式也要复杂得多。

③ 控制报文：分布式数据库系统中，除了数据报文外，还增加了控制报文。因为除了要对数据进行存取操作外，还要对各子事务的操作进行协调，这样就有了大量的控制报文要在网上传输。

（2）分布式数据库故障处理的的原则

故障的发生会影响数据库中数据的正确性，甚至破坏数据库，从而影响数据库系统的可靠性和可用性。因此，数据库管理系统都对故障恢复机制狠下功夫，认真地进行研究和开发。研究数据库系统中故障的恢复，主要是指如何恢复因故障而破坏的数据库，使数据库恢复到正确状态。由前面分析可知，在分布式数据库系统中，故障的类型很多。当发生事务故障时，保证事务原子性的措施称为事务故障恢复，简称为事务恢复。事务本身的故障和系统的故障是造成数据库完整性和一致性被破坏的主要原因。事务恢复主要是依靠日志来实现的。恢复应遵循的原则如下。

① 孤立和逐步退出事务的原则

对于不影响其他事务的可排除性局部故障，例如事务操作的删除、超时、违反完整性规则、资源限制、死锁等，应令某个事务孤立地和逐步地退出，将其所做过的修改复原，即执行 UNDO。

② 成功结束事务原则

成功结束事务所做过的修改应超越各种故障，当故障发生时，应该重新执行（REDO）事务的所有操作。

③ 夭折事务的原则

若发生了非局部性的不可排除的故障，例如系统崩溃，则撤销全部事务，恢复到初态。

这有两种做法，一种是利用数据库的备份实现；另一种是按反向顺序操作，复原其启动以来所做过的一切修改。

从集中式事务恢复可以了解事务恢复的一般过程，对分布式事务来说，由于处于网络环境下，其恢复处理远比集中式事务恢复要复杂得多。在分布式事务恢复中，本地事务的恢复和集中式事务的恢复相同，由本地事务管理器（Local Transaction Management，LTM）具体执行；而整个分布式事务的恢复由分布式事务管理器（Distribute Transaction Management，DTM）与 LTM 协同完成。

（3）分布式事务的两段提交协议

分布式 DBMS 的事务恢复要比集中式 DBMS 复杂得多。对于集中式 DBMS，为了在故障发生时提供必要的信息用于进行事务恢复，事务正常执行时需要执行某些特定的操作。需要在每个场地都对日志进行维护。在分布式 DBMS 中，除了集中式 DBMS 维护的信息外，提交协议的所有操作也都进行记录。

目前使用最为广泛，且已经成为工业标准的提交协议是**两阶段提交协议**（Two Phase Commitment Protocol，2PC），它把原本属于集中式 DBMS 的本地原子性提交行为的效果扩展到分布式 DBMS 事务，保证了分布式事务提交的原子性，并在不损坏日志的情况下，实现快速故障恢复，提高分布式数据库系统的可靠性。事务正常执行时，每个场地都对日志进行维护，并且在执行场地记录子事务的所有操作，除了在以前章节中描述的通常的日志行为外，还需要记录提交协议的相关信息，来确保给定事务的所有子事务要么全部提交要么全部中止。事务场地的事务管理程序称为事务的协调者（Coordinator），子事务执行场地的事务管理程序称为参与者（Participant）（必须遵从事务协调者的指令）。

只有协调者才拥有提交或撤销事务的决定权，而其他参与者各自负责在其本地数据库中执行写操作，并向协调者提出撤销或提交事务的意向。一般一个站点唯一地对应一个子事务，如果某一参与者与协调者在同一站点，虽然它们不需要使用网络进行通信，但仍在逻辑上认为它与协调者不在同一站点。图 6-5 描述了协调者和参与者的关系。

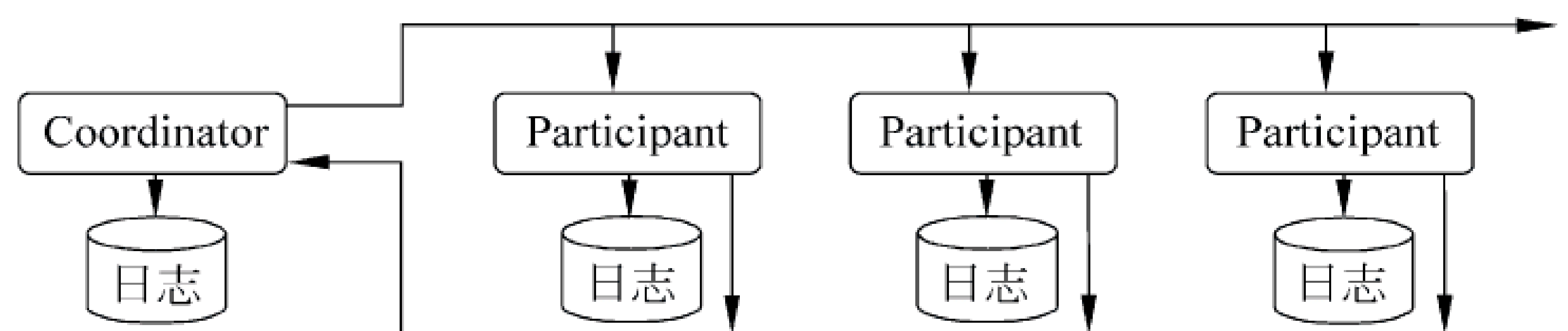


图 6-5　协调者和参与者的关系图

2PC 保证分布式事务提交的原子性，这是通过在分布式事务的结果生效以前，所有参与执行分布式事务的站点都同意提交而做到这一点的。这种同步的必要性有很多理由，如果某个事务正在读一项由另一个还未提交的事务更新的数据项的值时，相应的参与者就不会同意马上提交该事务。另一种参与者不同意提交的可能的原因是发生了死锁，这要求某

一个参与者撤销事务。注意，参与者不需要任何其他进程来通知就可以撤销一个事务，这种能力相当重要，称为单方面撤销。

2PC 把事务的提交过程分为两个阶段。

第一阶段是表决阶段，目的是形成一个共同的决定。开始时，协调者在它的日志中写入一条开始提交的记录，再给所有参与者发送“准备提交”消息，并进入等待状态。当参与者收到“准备提交”消息后，它检查是否能提交本地事务。如果能提交，参与者在日志中写入一条就绪记录，并给协调者发送“建议提交”消息，然后进入就绪状态；否则，参与者写入撤销记录，并给协调者发送“建议撤销”消息。如果某个站点做出“建议撤销”提议，由于撤销决定具有否决权（即单方面撤销），发出“建议撤销”的站点就可以直接忽略这个事务。协调者收到所有参与者的回答后，它就做出是否提交事务的决定。只要有一个参与者建议撤销，协调者就必须从整体上撤销整个分布式事务，因此它写入一条撤销记录，并给所有参与者发送“全局撤销”消息，然后进入撤销状态；否则，它写入提交记录，给所有的参与者发送“全局提交”消息，然后进入提交状态。

第二阶段是执行阶段，目的是实现这个协调者的决定。根据协调者的指令，参与者或者提交事务，或者撤销事务，并给协调者发送确认消息。此时，协调者在日志中写入一条事务结束记录并终止事务。图 6-6 描述了两阶段提交协议的参与者和协调者的交互。

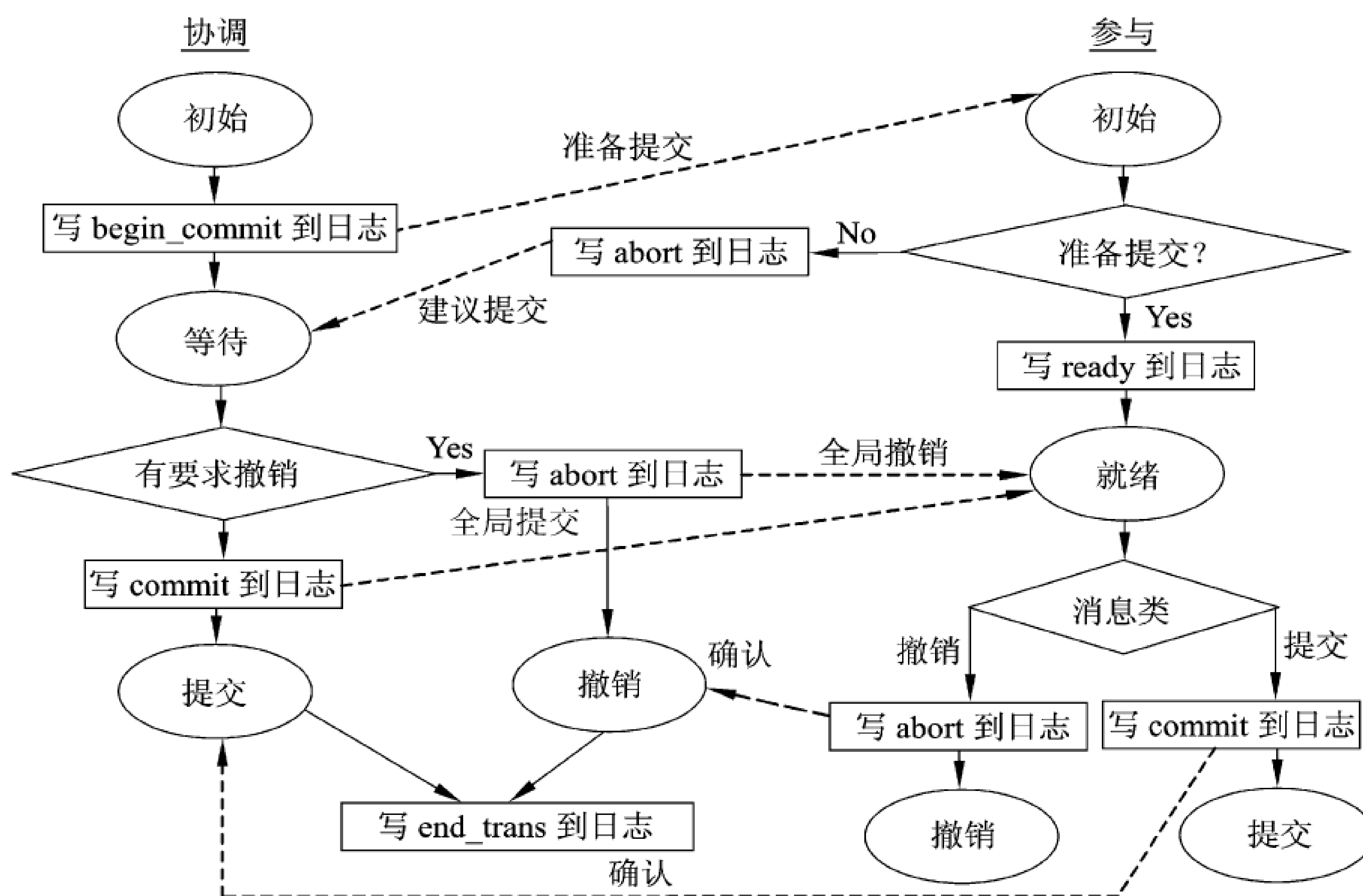


图 6-6　两阶段提交协议活动图

请注意协调者做出事务的全局终止决定的方式，该决定受两条规则的支配，这两条规则称为全局提交规则。

① 只要有一个参与者撤销事务，协调者就必须做出全局撤销决定；

② 只有所有参与者都同意提交事务，协调者才能做出全局提交决定。

从图6-6中可以看出以下一些关于两阶段提交协议的一些重要之处。

① 两阶段提交协议允许参与者可以单方面撤销事务；

② 一旦参与者确定了提交或撤销提议，就不能再更改它的提议；

③ 当参与者处于就绪状态时，根据协调者发出的消息的种类参与者可以转换为提交状态或撤销状态；

④ 协调者依据全局提交规则做出全局终止决定；

⑤ 注意协调者和参与考可能进入某些相互等待对方发送消息的状态。为了确保它们能够从这些状态中退出并终止，要使用定时器。每个代理进程进入一个状态时都要设置超时时间。如果所期待的消息在定时器超时之前没有到来，定时器向代理进程报警，进程根据超时协议执行相应动作。

（4）分布式事务的两段提交协议对故障的恢复

根据难点分析中总结出的关于分布式事务可能发生故障的分类，可以将故障恢复的策略对应分成场地故障恢复、网络传输中报文丢失故障恢复和网络分割故障恢复，并予以详细介绍。

① 场地故障恢复

当某个场地在系统崩溃后重启时，系统将调用恢复进程，读系统日志并处理所有在系统崩溃时正在执行的事务。某个场地的事务管理程序可能作为某些事务的协调者，同时又作为其他事务的参与者。在恢复过程中根据故障发生的不同状况可能需要下面的操作。

- 当一参与者在写入“建议提交”前发生故障时，该参与者无法向协调者发回应答信息，因此，当协调者等待超时后，将决定终止事务。当该故障恢复后，该参与者无须收集其他场地的信息即可终止事务。
- 参与者进程在写入“建议提交”后发生故障，这时其他的参与者可以正常地结束该事务，“提交”或“撤销”，因为协调者可以根据收到该参与者的应答决定“提交”或“撤销”。因此，故障恢复后，该参与者要访问协调者或其他参与者，以了解协调者对事务做出的决定，然后执行相应的操作“提交”或“撤销”。这里假设在日志中写入“建议提交”记录和发送“建议提交”信息给协调者。这两个动作具有原子性，要么都执行，要么都不执行。
- 协调者在日志中写入“准备提交”记录后，写入“全局提交”或“全局撤销”前发生故障，这时已发出“建议提交”信息的参与者等待协调者恢复。协调者的重启过程从头恢复提交协议，从“准备提交”记录中读出参与者的标识符，重发“准备提交”报文给参与者，重新执行提交过程。
- 协调者在写入“全局提交”或“全局撤销”记录以后，在写入“事务结束”记录以前发生故障。在这种情况下，协调者恢复时必须给所有的参与者重发其决定，

未收到信息的参与者不得不等待协调者的恢复。

② 报文丢失故障

- 至少有一个参与者的应答报文（“建议提交”或“建议撤销”）丢失了。在这种情况下、协调者将等待应答而超时，整个事务被撤销。这种情况只由协调者发现，但它无法决定是场地故障还是通信故障，而参与者能够正确执行，它不会启动恢复过程。
- 丢失“准备提交”报文，由于至少有一个参与者收不到“准备提交”命令，因此参与者处于等待状态，而协调者也等待参与者的应答，所以协调者会因为等待超时而撤销事务。这种情况和上述一样。
- 丢失“全局提交”或“全局撤销”报文，这种情况下参与者处于等待协调者命令的状态下，当参与者未收到命令时，会因等待而超时，这时向协调者请求重发该命令的信息。
- 丢失了“确认”报文，当协调者未收到全部参与者的“确认”报文时，协调者会因等待而超时，这时协调者重发命令报文给参与者，参与者必须给予“确认”报文应答，即使此时相应的子事务已不在活动也要重发。

③ 网络分割故障

假设在出现网络分割时，整个网被分为两个组，包含协调者的组称为协调者组，其他的则组成参与者组。这种情况对于协调者来说相当于参与者组中的多个参与者同时发生故障，这时协调者可以做出决定，然后把命令发给协调者组中的参与者，因此这些场地上的子事务可以结束。而对于失去联系的参与者，它们则认为协调者出现故障，根据它们所缺少的应答信息，进行相应的故障处理。

（5）分布式数据库事务的三段提交协议

事务的阻塞是指一个场地的子事务本来是可以执行结束的，然而由于分布式数据库的故障，它必须等待故障恢复以后得到需要的信息后才可以做出决定，而故障情况是不可以预料的，该子事务又占有一些系统资源不能释放，无法继续执行，这时称之为事务进入阻塞状态。

事务出现阻塞的原因可能很多，例如：当参与者等待协调者的应答时，可能因为网络故障或协调者故障使之收不到应答信息而出现等待超时，这时事务进入阻塞状态，重发“建议提交”信息，要求协调者给予应答，直到网络故障或协调者恢复并给予应答，参与者才做出决定继续执行（提交或撤销），若一直收不到应答，则事务一直处于阻塞状态而挂在相应的场地上，因此，阻塞降低了事务的可用性。

如何使一提交协议成为非阻塞的提交协议呢?在 2PC 协议中，参与者的提交是在它知道了其他所有的参与者均发出了“建议提交”的报文以后进行的。若在 2PC 中增加一段使得参与者的提交不仅要等到它知道所有的参与者均发出了“建议提交”的报文，而且还知道所有参与者的状态（如它们是处于故障状态，还是已经恢复）以后才执行。这时 2PC 即变成 3PC 协议，即三阶段提交协议。在 3PC 协议中，报文有 3 次接收和发送，协调者第 2 次向参与者发出的报文不是“全局提交”报文，而是提交前的“全局预提交”报文，告诉

所有的参与者均可以进入准备提交状态，而参与者的应答也不是提交子事务，而是发出“准备就绪”报文。在第三阶段中，当协调者收到全部的“准备就绪”的应答时才向所有的参与者发“全局提交”报文。此时，所有的参与者均知道其他的参与者已经进入“准备提交”状态。达到这一点，每个参与者均可以自己做出决定，撤销或提交，而不必因等待协调者的应答而进入阻塞状态，因为即使此时发生故障，系统的恢复机制迟早会恢复到故障前一刻的状态，即各参与者的子事务总会提交。因此，参与者可以自行决定先执行下去而不是处于等待状态，从而减少了阻塞。3PC 协议的工作过程如下。

第一阶段，协调者向所有的参与者发“准备提交”报文，由每个参与者据自己的情况进行投票，只有所有的参与者应答“建议提交”才进入第二阶段；

第二阶段，协调者向所有的参与者发“全局预提交”报文，参与者收到该报文后若已经准备好提交，则应答“准备就绪”报文，否则进行撤销处理；

第三阶段，协调者收到所有的参与者“准备就绪”应答后，就向所有的参与者发“全局提交”报文，此时每个参与者都知道其他的参与者赞成提交，因此它可以收到“全局提交”报文后进行提交。

3PC 可以避免阻塞是基于一定的故障模型的，如果发生了网路分割故障，采用 3PC 协议同样存在问题，没有一种协议能够解决所有的故障，3PC 协议仅仅是降低了阻塞发生的可能性，但不是完全的非阻塞协议。

（6）分布式数据库事务的三段提交协议的恢复

由于系统的故障，报文可能没有收到。与 2PC 一样，3PC 也采用超时方法处理。

① 协调者没能及时发出“准备提交”报文，导致参与者等待超时，这时参与者决定撤销。

② 协调者等待参与者投票结果超时，这时协调者决定撤销。

③ 参与者处于“赞成”提交状态，等待“全局预提交”命令时出现超时，这时进入恢复处理过程。

④ 参与者处于“准备就绪”状态等待协调者的“全局提交”命令出现超时，也进入恢复处理，这时只要有至少一个参与者处于活动状态，子事务就不会阻塞。因为恢复后的参与者可以从活动子事务得到有关提交处理的信息，从而得知协调者的决定，依据协调者的决定确定自己应做的处理。

事务在进入恢复前有下述的两种状态是不相容的。

① 一个参与者在其他任何一个活动的参与者处于赞成提交状态时，不可能进入“提交”状态。

② 一个参与者在另一个参与者进入提交状态或任何一个参与者都进入准备就绪状态时不能进入撤销状态。

因此，恢复时参与者可以根据活动事务的状态决定相应的处理办法。在 3PC 协议中，恢复机制唯一可以做的是就近访问一个活动的参与者，确切地说是访问在它进入恢复处理

前最近的活动参与者。如果所有的参与者处于“赞成”提交或“撤销”状态，则肯定没有一个参与者已提交，因此可以通知全部参与者“撤销”。如果已经有一个参与者“准备就绪”或“提交”，则肯定没有一个参与者被“撤销”，所以可以通知全部参与者提交，在通知“全局提交”前应先使仍处于“赞成”提交的参与者进入“准备提交”状态，然后再进入“提交”状态。因为在通知提交前，任何一个参与者均可能再发生故障，所以应避免其中一个参与者处于“赞成”提交状态而另一个处于“提交”状态。

6.1.5　分布式数据库的应用

由分布式数据库系统特点就可知，在一个各场地有数据库的计算机网络系统中，若有分布式数据库管理系统，则可使系统中的数据库实现共享，利用率高，存取快，可靠性高，用户使用这些数据库如同使用本地数据库一样，极为方便。反之，若没有分布式数据库管理系统，只利用现有的一般网络操作系统来存取网络中的数据，凡使用过这类系统的用户都会感到相当麻烦和困难，而且这类系统在有一定数据冗余（这是获得可靠性所需要的）的情况下很难保证数据的一致性，故障点的自动切除和故障后的恢复以及并发控制等问题就更没有保障了。所以，研究各种性能优良的分布式数据库系统，无论是同构还是异构型，都是极为必要的。

分布式数据库系统将广泛用于各企事业单位的人事、财务和库存控制等管理信息系统，百货公司销售点的经营销售信息处理系统，电子银行等的在线处理系统，国家政府部门的经济信息系统，大规模数据资源如人口普查、气象预报，环境污染，地震监测等数据库系统，军事指挥控制系统，具有多数据处理中心的企业报告生产系统，医院的病历药品管理以及辅助诊断、病人监护系统，工业控制和管理的集散型系统，计算机综合自动化制造系统以及各种办公自动化系统等。

6.2　网络环境下数据库系统的设计与实施

网络环境下数据库系统的设计与实施的知识框图如图 6-7 所示。

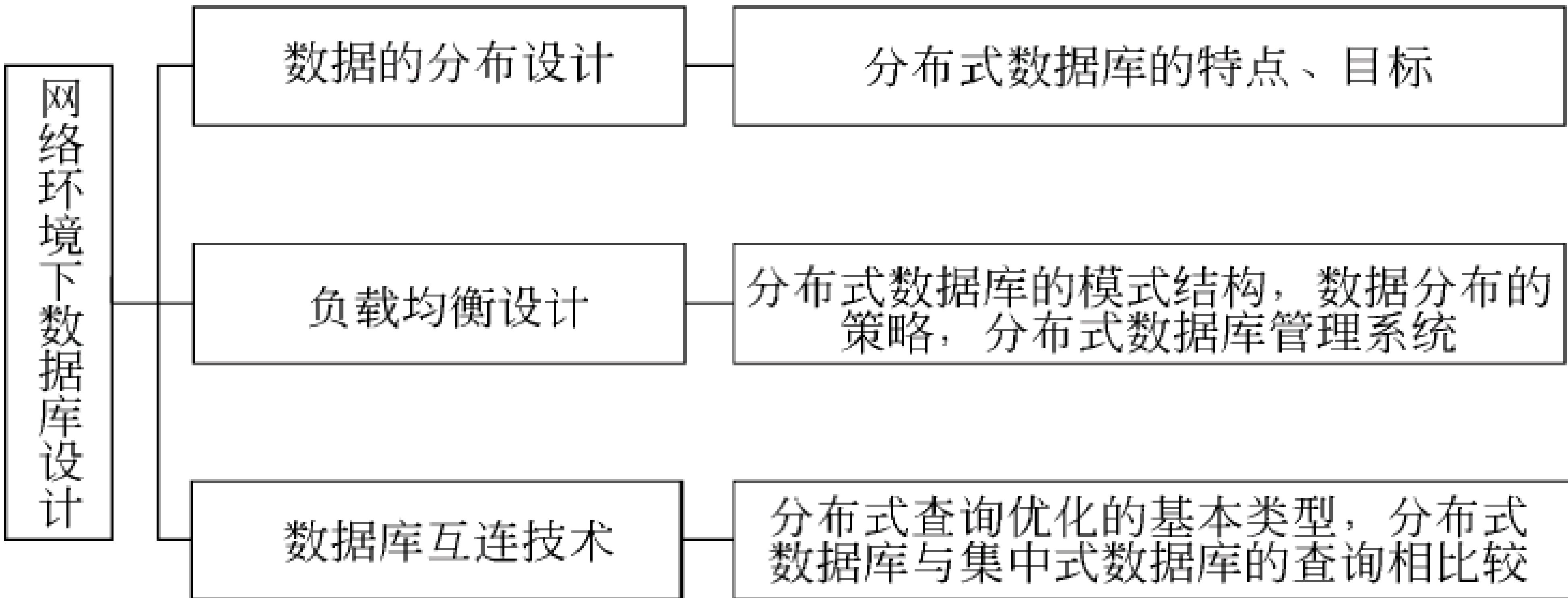


图 6-7　网络环境下数据设计基础知识框图

6.2.1　数据分布设计

分布式数据库是在计算机网络的基础上发展起来的，现在已经成为信息处理中的一个重要领域。分布式数据库的本质应用就是对分布于网络中各节点的数据进行各种不同要求的处理。

无论是出于对经济的考虑，还是出于对数据存储、处理的性能要求，数据应该如何分布，如何进行负载均衡，始终是人们所关心的一个重要问题。因而数据分布设计一直是分布式数据库系统设计中一个主要方面。

（1）数据分布设计的目标

① 处理的本地性：对数据的处理主要指查询操作和更新操作。实现本地性的目的是为了减少远程访问的次数，以节省通信费用。

② 分布数据的可用性和可靠性：即在正常条件下应该存取的副本不能使用时，可以使用另外一个副本；在某一节点出现故障或某一个副本遭到破坏时，可以使用另外一个副本进行恢复。

③ 工作负载的分布：其目的是利用每个站点计算机的不同能力和使用率，提高执行的平行程度。

④ 存储的费用和可用性：即保证在一定的可用性的前提下，尽可能地降低存储费用和灾难恢复的代价。

（2）数据分布设计的方法

① 集中式数据分布方法：集中式就是把数据库完整地存放在一个节点。因为仅仅有单一的数据库副本，只要存放数据库的节点有故障或者数据库遭到破坏，就可能会引起无法恢复的灾难后果。

② 重复式数据分布方法：指在每个节点都存放数据库的完整副本。因为每个节点都存放了数据库的完整副本，所以只要还有一个节点是正常的，数据库就是有效的，并且可以由此对已经遭到破坏或已经出现故障的节点进行恢复。

③ 分割式数据分布方法：分割式是根据对某些数据的大量操作大多来自于某些（个）特定节点的实际应用状况，把数据库分割成无重叠的很多部分，并分别存放在不同的站点。如果某一节点出现故障，则存储于此节点的数据就失效，但存储在其他节点上的数据仍旧可用。

④ 分段式数据分布方法：分段式是在分割式、重复式和集中式的基础上，吸取各种方法的优点综合而成的一种数据分布的设计方法。其主要设计思想是：把数据库在逻辑上分割成若干个不重叠的部分(即分割式)，在物理上把这些部分分配到一个或几个节点之中。如果某一个节点仅存有一个副本，当此节点出现故障时，这些唯一的数据就失效了，但存储于其他节点的数据以及存储在此节点但在其他节点但数据副本的数据仍然可以继续使

用，从而使得数据库仍旧能够维持工作。

6.2.2　负载均衡设计

在网络环境下的应用系统中，服务器必须具备提供大量并发访问服务的能力，其处理能力和 I/O 能力已经成为提供服务的瓶颈。显然，单台服务器有限的性能不可能解决这个问题，一台普通的服务器的处理能力或许可以达到每秒几万到几十万次的请求，却无法处理短时期骤增的几百万次或者更多的请求，但若是能够将多台这样的服务器组成一个系统，并通过软件技术、网络支持、硬件支持等手段将所有的请求处理合理地分配给所有的服务器，那么这个系统就能顺利响应短时间内几百万次的骤增并发处理请求，这就是负载均衡设计的基本思想。

（1）负载均衡设计目标

负载均衡（local balancing）是一种策略，它建立在网络环境下，能够让多台服务器或者多条链路共同承担一些繁重的的请求处理，从而提供一种廉价有效的方法来扩展服务器带宽和增加处理吞吐量，加强整个系统数据处理能力，提高系统使用的灵活性和可用性。

（2）负载均衡设计方法

① 静态负载均衡：就是事先确定好系统内各个任务的分发策略。

② 动态负载均衡：通过一些工具实时地分析数据包，掌握系统中数据库服务器的负载及数据流量状况，把任务合理分配出去。

（3）静态负载均衡设计

静态负载根据事先确定好网络任务的分发策略，运行时负载不能够重新分配，其目标是调度一个任务集合，使它们在各个目标节点上有最小的执行时间。静态负载均衡因此又被称为调度问题。

设计调度策略有 3 个主要因素。

① 网络互联拓扑

网络互联拓扑分为静态和动态，静态网络由点到点直接连接，并且进程执行的过程中不能改变。动态网络由交换信道实现，能够根据用户程序规定动态配置满足通信需要。可以在系统搭建的初始阶段，由程序根据能够预见到的静态或动态的网络状况给出不同的数据库操作的执行方案。

② 任务划分（粒度决策）

任务划分的一个主要目标就是尽可能消除处理器之间通信引起的开销。

一个给定任务划分的粒度定义了任务分解中影响通信开销的所有单元的平均尺度。给定任务可以划分为细粒度、中粒度和粗粒度。如果数据单元（即粒度）太大，就会降低系统的并行度，而粒度太小又会引起进程切换和通信的开销增大。

③ 任务分配

任务分配就是在给定了明确网络连接方式的并行和分布式系统中合理有效分配、调度

既定的任务粒度。

（4）动态负载均衡设计

动态负载均衡：在网络环境下，由于任务经常体现出随机性，以及各节点处理能力有所差异，容易产生各节点负载处理不协调，空闲与超负荷运载的情况同时发生在同一个系统中。

如何有效地提高系统的资源综合利用率，减少任务的平均响应时间，这成为动态负载均衡产生的原因。负载均衡就是这样一种方法，它设法对分配给各节点的任务重新实时调度，可以通过进程迁移（又称任务迁移）使得各节点负载大致相等。

动态负载均衡算法的组成要素：

- 启动策略
- 转移策略
- 受益策略
- 位置策略
- 信息策略

6.2.3　数据库互联技术

WWW 技术自问世以来，以其特有的图文并茂、可视性强和交互方便等特点获得了迅猛发展。但 Web 的发展经历了两层体系结构和三层体系结构阶段。在两层体系结构阶段，Web 服务器和浏览器之间是一种基于 HTTP 的简单的信息发布/获取的两层客户机/服务器结构，服务器只提供信息存储和信息发布功能，用户通过浏览器被动地获取信息，功能单一，缺少交互性。随着分布对象技术的发展，出现了现在流行的三层体系结构，即客户端—应用服务器—数据库服务器三层体系结构。

（1）三层体系结构的概念

三层体系结构中，Web 浏览器作为客户层，提供图形用户界面。

负责与用户进行交互。它通过 HTTP 从应用层的 HTTP 服务器下载超文本页面，同时下载并执行内嵌在页面中的客户方程序或中间代码（如 Java 字节码）。这些客户方程序能通过内部通信机制向应用服务器中有关服务对象发出请求。服务对象封装了相关的业务逻辑，它们之间可通过内部协议彼此通信，并能访问资源层的数据库对象或其他的应用程序，以协同完成客户请求。由于采用了分布对象技术，增加了应用层，将客户与资源层分开，降低了 Web 服务器的负载，避免了 Web 服务器的性能缺陷对整体性能的影响，并具有连接缓冲、负载均衡、安全管理等功能，大大提高了 Web 应用整体的灵活性、可伸缩性和可扩展性。

（2）三层体系结构的基本框架

三层体系结构的基本框架如图 6-8 所示。

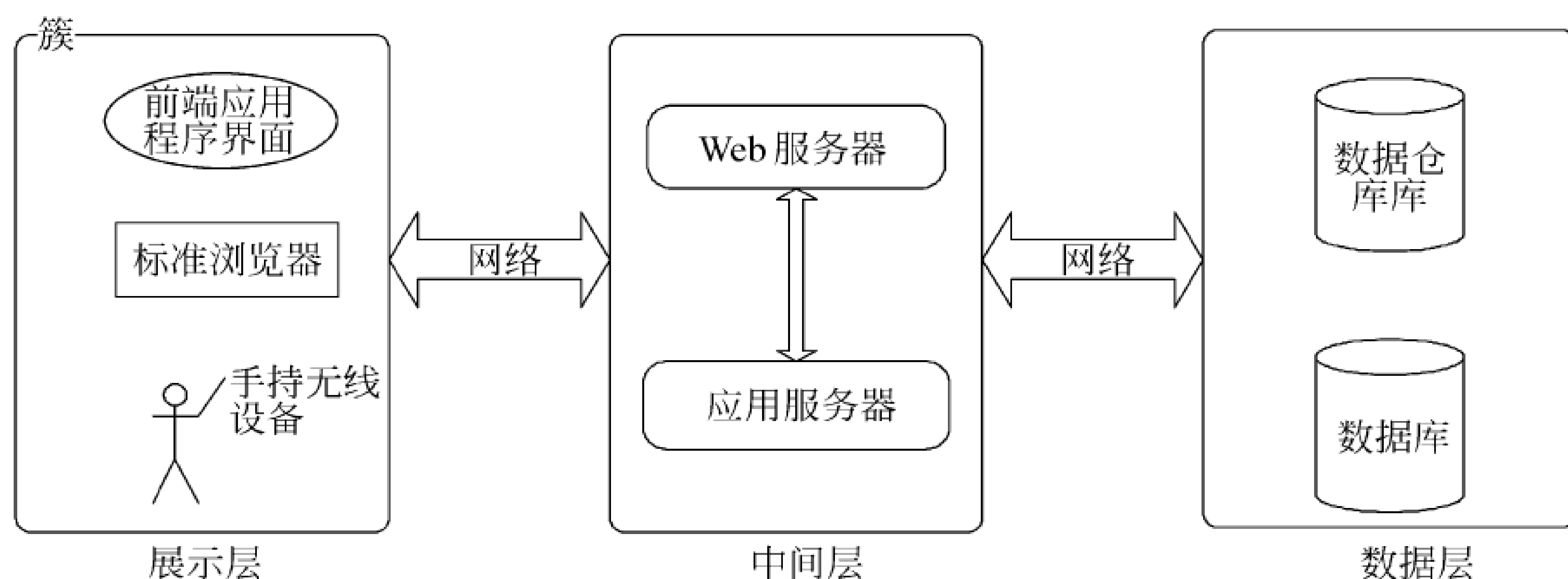


图 6-8　三层体系结构基本框架

（3）三层体系结构概述

① 展示层

用户需要一个自然的界面发出请求，提供输入并查看结果。Internet 的广泛使用使得基于 Web 的界面越来越流行。

② 中间层

某些展示逻辑从原先的胖客户端形式后移到 Web 服务器上执行，而很多的应用逻辑将在中间层的应用服务器上执行，一个企业级的中间层应该包含 Web 服务器和应用服务器，并提供高效的数据库访问方式。

③ 数据层

数据密集应用所涉及的关系数据库、数据仓库、文档型数据库等将在此处被配置。

（4）三层体系结构的优点

- 异构的系统：应用可以在不同层上利用不同平台和不同软件组件的长处，可以在任意一层上修改或替换代码，而对其他层没有过多影响。
- 瘦客户端：客户端只需要为展示层提供足够计算能力的环境即可，换句话说，就是应用对客户端的要求比较低，只需要可以运行例如 Web 浏览器的环境就可以了。
- 集成的数据访问：在很多应用中，数据必须从不同的数据源获得，这可以在中间层被透明地处理。在中间层，可以集中管理与所有数据库系统的连接，而展示给用户的确是统一的界面。
- 对多用户的可扩展性：在系统部署的过程中，完全可以按照现有系统客户端的访问需求配置硬件环境，后期随着客户访问的不断增加，通过集群技术、负载均衡技术等将适当地增加中间层服务器的处理能力，而这一切对前台的展示层和后台的数据层都是透明的。
- 软件开发的优势：通过将应用清晰地划分为展示、数据处理、业务逻辑 3 个部分，可以使软件开发变得更好控制。首先，业务逻辑是集中的，因此易于测试、维护、排错和修改。其次层与层之间的交互通过标准的 API 协议来进行，因此，应用的

每一层都可以建立在可重用的构件上，这些构件可以独立开发，随时更换，使对其他层的影响降到最低。

6.2.4 动态网页

在典型的三层体系结构的应用系统中，服务器对客户端提供信息的服务，并由后台数据库提供实时更新的数据，显然这样的页面处理功能相对于以前无数据交互的页面是动态的。用户可以根据自己的要求获取最新的、按客户需求定制的格式化信息，而且可以与服务器进行对话，进行信息交互。这要求 Web 系统在信息的组织管理中不能单单依靠 HTML 来显示内容，更多地要借助于可编程的控制性语言、脚本显示内容。

（1）动态网页的原理

动态网页的工作原理就是将服务器端（Web 服务器）的脚本结合 HTML 语言，在浏览器端生成动态交互的网页。其中服务器端脚本包括 JSP、ASP 等，同时也应用了 XML 技术，加强了 Web 页面数据的可扩展性。

（2）动态网页数据库访问机制

Web 页面与数据库地连接是动态网页的基本要求。目前基于数据库的动态 Web 网页数据库访问机制主要有两种类型：服务器端和客户端方案。服务器端方案实现技术有 CGI、SAPI、ASP、PHP、JSP 等；客户端方案实现技术有 JDBC（Java Database Connectivity）、DHTML（Dynamic HTML）等。其中 ASP 是微软开发的脚本语言技术，它嵌入在 IIS 中，因此 ASP 也就顺理成章地成为大部分 Windows 用户首选的脚本语言。

通常，动态网页运行的环境由硬件元素和软件元素组成。硬件元素包括 Web 服务器、客户机、数据库服务器、网络。软件元素包括客户端必须有能够解释执行 HTML 代码的浏览器（如 IE、Netscape 等）；在 Web 服务器中，必须具有能执行可以自动生成 HTML 代码的程序的功能，如 ASP、CGI 等；具有能自动完成数据操作指令的数据库系统，如 Access、SQL Server 等。

Web 服务器使用 HTTP 对客户机的请求给予应答。一个 Web 服务器在 Internet 上都有一个唯一的地址，这个地址可以是一个域名，也可以是 4 个以点分隔的 0 到 255 之间的数。例如 www.yahoo.com、202.106.168.67。如果客户机提出一个合法的请求，那么 Web 服务器就会把请求的内容传送给客户机。Web 服务器不仅能够传送各种文件，还可以传送某个程序的输出结果，这给 Web 和数据库的结合创造了条件。Web 服务器的种类很多，比较著名的有 IIS 和 Apache 等。

图 6-9 给出了典型的 Web 和数据库的运行模式。

在脚本程序中连接数据库一般都需要相应的接口来完成。连接数据库的常用方法有：ODBC、DAO、RDO 及 ADO 等。

① ODBC。ODBC（Open Database Connectivity，开放式数据库连接）是微软开发的一套统一的程序接口。通过这个接口可以访问不同厂商生产的数据库。经过多年的改进，

它已成为访问服务数据库的标准。事实上，ODBC 技术已成了后来 DAO、RDO 及 ADO 等数据库访问技术的基础。

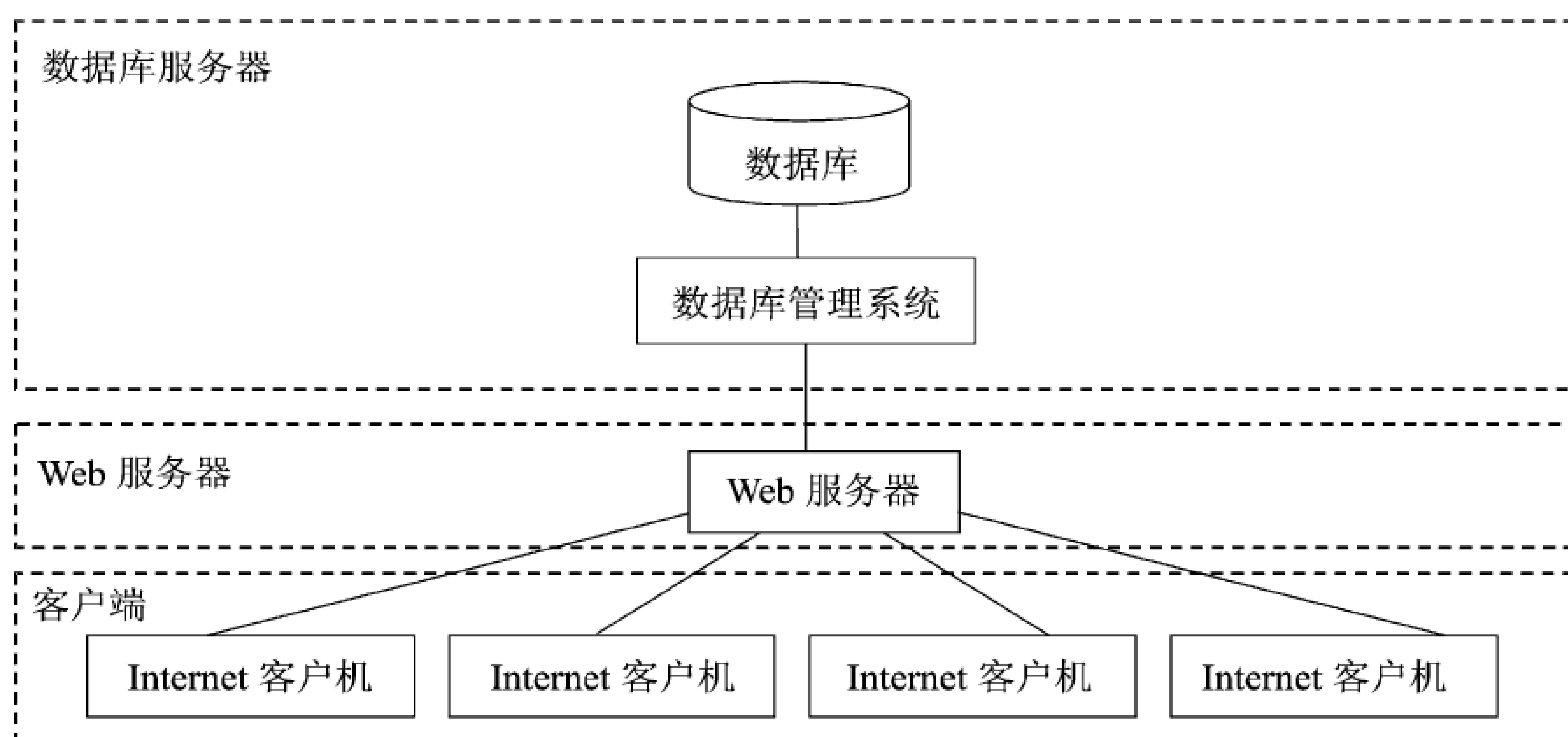


图 6-9　Web 和数据库的运行模式图

② DAO。DAO（Data Access Objects，数据访问对象）是微软公司开发的一套主要应用程序及开发工具，用它可以访问数据库的标准对象，如 Access、VB、Excel、Word 等。

③ RDO。RDO（Remote Data Objects，远程数据对象）是微软公司为增强 DAO 的功能而推出的新产品。该产品强化了 SQL Server 的访问功能，提高了它的执行效率。

④ ADO。ADO（ActiveX Data Objects，ActiveX 数据对象）是微软在 Internet 领域采取的新举措。它本身并不是一项新技术，从对象结构的角度来看，它比 DAO 提供的对象更少；从存取 SQL 服务器的角度来看，它提供的功能也不如 RDO。但它汲取了 DAO 和 RDO 最精华的部分，成为一个更适合于 Internet 的小而精的对象群。因此，ADO 实际上是脚本程序连接数据库的一种选择。

（3）动态 Web 网页技术的现状和发展趋势

由于 Web 应用开发的独特性，应用开发平台成为众多厂商的关注焦点。目前市场上存在很多的 Web 应用标准、集成开发环境。流行的主要是 ASP、PHP、JSP 三种。

① ASP

ASP（Activex Server Pages）是由微软创建的 Web 应用开发标准，ASP 服务器已经包含在 IIS 服务器中，ASP 服务器将 Web 请求转入解释器中，在解释器中将所有 ASP 中的脚本进行分析，然后执行，同时可以创建 COM 对象以完成更多的功能，ASP 中的脚本是 VBScript。优点：安装配置方便，开发简单，易学；开发工具功能强大。不足：ASP 使用了组件，因而将导致大量的安全问题；无法实现跨平台，只能应用于 Windows NT/2000。

② PHP

PHP 由于其良好的性能及免费的特点，现已成为互联网中应用非常流行的一种应用开

发平台。

优点：简单易学、跨平台、有良好数据库交换能力的开发语言；与 Apache 及其扩展库紧密结合；良好的安全性。不足：安装配置复杂；缺少企业级的支持；作为自由软件，缺乏正规的商业支持；无法实现商品化的商业开发。

③ JSP

优点：可移植性好，支持多种平台；强大的可伸缩性；多样化与强大的工具支持。不足：安装配置、管理较为复杂；运行速度较慢。建议开发大型应用系统采用 JSP。

6.2.5　动态网页的具体应用

（1）CGI 的应用

公共网关接口（Common Gateway Interface，CGI）是最早出现的动态发布网页技术，由于其开发较早，技术成熟，因此目前仍是动态网页开发的主力之一。Common 表示确保 CGI 可以使用多种程序语言和多种不同的系统交互，Gateway 表示 CGI 的力量不在于它本身所做的事，而在于它提供了连接其他系统的潜力，例如数据库和图形生成工具等，Interface 表示 CGI 对如何更好地利用其特性提供了明确的定义，换句话说，可以设计程序来适当利用这个接口。CGI 是 Web 服务器调用外部程序的接口。通过 CGI，Web 服务器能完成一些本身所不能完成的工作。早期很多著名的服务器都以自己独特的方式，支持服务器端的可执行程序，用来帮助完成客户机的请求。为某个服务器写的程序要在其他服务器上使用时，就必须做较大的修改，原因是每个服务器与可执行程序之间传递信息的内容和方式都不尽相同。为此就形成了一个公共标准 CGI，使得为一个服务器写的程序能够在任何服务器上运行。通过这个公共网关接口，服务器可以向 CGI 程序发送信息，CGI 程序也可以向服务器发送信息。可以使用 C Shell、Perl、C、C++、Fortran 和数据库语言等任何能够形成可执行程序的语言编写。

如果现在要让 Web 服务器与其他系统结合，比如后台数据库系统，则 CGI 程序会起到程序接口的作用，将接收到的参数进行预处理，转换成所要结合的数据库系统能够识别的形式，对于数据库系统而言，常常就是指数据库系统能够识别运行的标准的 SQL 语句。当其他系统完成数据处理后，如果有结果返回，则 CGI 程序获得并处理其他系统所传回的数据，然后将其按一定的标准格式再送回至 Web 服务器，由 Web 服务器以网页的形式传回到客户端。为了灵活使用各种数据库系统，CGI 程序支持 ODBC 方式。CGI 程序不直接访问数据库系统，而是通过 ODBC 数据库接口管理器实现。应用程序以标准 SQL 语句访问 ODBC，由 ODBC 用不同的数据库所提供的 ODBC 驱动程序将 SQL 语句转换成本数据库所能执行的语言，然后访问数据库。当数据库将结果返回 ODBC 时，ODBC 同样将返回结果进行预处理，以标准形式返回给 CGI 程序。这样使用 ODBC 方式访问数据库的优点是程序员在开发系统时不必考虑后台数据库的类型，只要以标准 SQL 语句编写数据库查询语句访问 ODBC 数据库接口，由 ODBC 来负责对各种数据库的支持。不论是使用大型数据

库，还是小型数据库，开发人员都不必更改 CGI 程序。这样就给系统的开发、维护和升级都带来很大的方便和灵活性。

（2）ASP 的应用

ASP，全称 Active Server Page，它提供了一个在服务器端执行脚本指令的环境（包括 HTML、VBScript 和 JavaScript 等），通过这种环境，用户可以创建和运行动态的 Web 应用程序。由于所有的程序都在服务器端执行，这样就大大减轻了客户端浏览器的负担，提高了交互速度。利用 ASP 不仅能够产生动态的、交互的、高性能的 Web 应用程序，而且可以进行复杂的数据库操作。ASP 本身包含了 VBScript 和 JavaScript 的引擎，使得脚本可以直接嵌入 HTML 中，而且还可以通过 ActiveX 控件实现更为强大的功能。

确切地说，ASP 并不是一种语言，它所使用的语言通常是 VBScript 或者 JavaScript，通过这两种脚本语言能够很方便地开发 ASP 应用。但决不能将 ASP 与 VBScript 或者 JavaScript 等同起来，VBScript 和 JavaScript 之间最大的区别就是它们的结构。VBScript 是 Visual Basic 的子集，如果曾经用过 Visual Basic 或者是 Visual Basic for Applications（VBA），对此就会觉得非常熟悉。不过它们并不是完全一样的，因为 VBScript 是特意为在浏览器中进行工作而设计的，它不包括一些在脚本这个范围以外的特性，如文件访问和打印等等。JavaScript 是从一组编程语言如 C、C++以及 Java 等中脱离出来的。如果以前曾经用过 C 或者是 Java，那么 JavaScript 的结构会觉得非常熟悉。但是，JavaScript 和 Java 是完全不同的两种语言。Java 是一种对于网页应用程序和非网页应用程序都可以使用的完全成熟的开发语言，而 JavaScript 是一种主要用于脚本编写的脚本语言。

ASP 能够提供 6 个内建对象，能够很方便地实现状态保存功能，可以很容易地从客户浏览器获取信息，并向浏览器反馈信息，因此，就能够很方便地运用 ASP 开发 Web 应用。

ASP 特点有以下几个特点。

① ASP 无须编译，ASP 脚本集成于 HTML 中，无须编译或连接即可直接解释执行。

② ASP 易于生成。使用常规文本编辑器即可进行页面的设计。

③ ASP 独立于浏览器。用户端只要使用可解释常规 HTML 码的浏览器，即可浏览 ASP 所设计的主页。

④ ASP 脚本是在站点服务器端执行的，因此，若不通过从服务器下载来浏览 ASP 主页，在浏览器端将看不到正确的页面内容。

⑤ 在 ASP 脚本中可以方便地引用系统组件和 ASP 的内置组件，还能通过定制 ActiveX 服务器组件来扩充功能。与任何 ActiveX Scripting 语言兼容。

⑥ 源程序码不会外漏。ASP 脚本在服务器上执行，传到用户浏览器的只是 ASP 执行结果所生成的常规 HTML 码，这样可保证程序代码不会被他人盗取。

ASP 所完成的功能如下。

① 处理由浏览器传送到站点服务器的表单输入。

② 访问和编辑服务器端的数据库表。使用浏览器即可输入、更新和删除站点的数据

库中的数据。

③ 读写站点服务器的文件，实现访客计数器等功能。

④ 取得浏览器信息管理等内置功能。

⑤ 由 Cookies 读写用户端的硬盘文件，以记录用户的数据。

⑥ 可以实现在多个主页间共享信息，以开发复杂的商务站点应用程序。

⑦ 使用 VBScript 或 JavaScript 等简易的脚本语言，结合 HTML，快速完成站点的应用程序。通过站点解释器执行脚本语言，产生或更改在客户端执行的脚本语言。

⑧ 扩充的能力强，可通过使用多种程序语言制作的 ActiveX Server Component 来满足自己的特殊需要。

ASP 是通过一组称为 ADO（ActiveX Data Objects）的对象模块来存取数据库的，无论采用的是什么数据库，只要该数据库具有对应的 ODBC 或 OLE DB 驱动程序，ADO 对象就能加以存取。事实上，ASP 提供的 ADO 对象模块包含了下列 6 个对象和 3 个集合，比较常用的是 Connection、Recordset、Command、Field 等对象。

Connection 对象：打开或关闭数据库连接。

Recordset 对象：存取表的记录，包括读取、插入、删除或更新表的记录。

Fields 集合：Recordset 对象所包含的 Field 对象的集合。

Field 对象：用来表示表的某一条记录。

Command 对象：执行查询并返回条件符合的记录（返回值为 Recordset 对象）。

Parameter 集合：Command 对象所包含的参数集合。

Parameter 对象：用来表示 Command 对象所需要的某个参数。

Errors 集合：某个方法调用失败所产生的错误集合。

Error 对象：用来表示方法调用失败所产生的某个错误。

ASP 的工作流程：当客户端的 Web 浏览器访问某一 Web 站点时，浏览器将 URL 发送给 Web 服务器请求信息。Web 服务器返回 HTML 页面响应。HTML 页面是已经格式化并存储在 Web 节点中的静态页面，也可以是服务器动态创建以响应用户所提供信息的页面，或者是列出 Web 节点上可用文件和文件夹的页面。

如图 6-10 所示，当用户申请一个 ASP 主页时，Web 服务器响应该 HTTP 请求，解释被申请文件。当遇到任何与 ActiveX Scripting 兼容的脚本（如 VBScript 和 JavaScript）时，ASP 引擎会调用相应的脚本引擎进行处理。若脚本指令中含有访问数据库的请求就通过 ODBC 与后台数据库相连，由数据库访问组件 ADO 执行访问数据库的操作。ASP 脚本是在服务器端解释执行的，它依据访问数据库后返回的结果集自动生成符合 HTML 语言的主页，去响应用户的请求。所有相关的工作都由 Web 服务器负责。

在结构关系上，ASP 是通过 ODBC 与数据库打交道。因此，向上可兼容各类数据库系统。而对于下层，ASP 产生的 HTML 对客户端的浏览器又有广泛的适应性。但 ASP 对 Web 服务器本身有所挑剔，这看起来似乎是一种缺陷，而实际上也许是一种商业策略——它只

支持微软各种操作系统下的 Web 服务器。

图 6-10 表示了 ASP 的工作原理。



图 6-10　ASP 工作原理图

（3）Servlet 和 JSP 的应用

Servlet 是 Web 组件程序，它可以动态地生成 Web 内容，支持 Web 应用的 HTTP 使用请求-响应机制。服务器接收请求、处理请求并返回适当的响应。Servlet 用面向对象的方式对这一过程建模，使用户能编写代码处理客户的请求并动态地响应。例如，Servlet 可能从一个表单读取数据并用它更新公司的订单数据库。Servlet 技术是通过动态 HTML 页面扩展 Web 服务器的一种强有力的方法。一个 Servlet 就是一个运行在 Web 服务器中的 Java 程序，Servlet 从浏览器中获取一个 HTTP 请求，生成动态内容（例如查询一个数据库），并把 HTTP 的响应返回给浏览器。

在 Servlet 之前，CGI 技术被用在动态内容中。然而，由于它的结构以及可升级性的限制，CGI 最后被证明为是不太理想的解决方案。

Servlet 技术在可升级性上有了很大的改善，它提供了公认的 Java 平台扩展、安全性以及强壮性等方面的优点。

Servlet 能使用所有的标准 Java APIs，在 Java 领域中，Servlet 技术为密集型应用程序（比如访问一个数据库）提供了很多的便利。便利之一就是 Servlet 运行在服务器端，服务器端具有多种资源且属于一个相对强壮的机器，因此占用客户端的资源相当少。另外一个便利就是 Servlet 在访问数据时更加直接。因为运行 Servlet 的 Web 服务器或者数据服务器在数据被访问时是与网络防火墙在一端的。

JSP 技术由 Sun 公司提出，利用它可以很方便地在页面中生成动态的内容，使网络应用程序可以输出多姿多彩的动态页面。JSP 技术通常与 Java Servlet 技术相结合，可以在 HTML 页面或者其他标记语言中内嵌 Java 代码段并且调用外部 Java 组件。它作为一个前端处理工具，可以使用 JavaBeans 实现复杂的商业逻辑和动态功能。

JSP 代码与 JavaScript 等网页脚本语言是不同的，在标准的 HTML 页面中可以出现的任何内容都可以在 JSP 页面中出现。

在一个典型的数据库应用中，JSP 页面将会调用某些 JavaBean 组件，这些组件可以通过 JDBC 或者 SQLJ 直接或间接地访问数据库。

JSP 页面在运行之前要被解释成 Java Servlet，解释过程是按需进行的，有时可能会提前进行，然后处理 HTTP 请求并生成响应信息，JSP 技术为编写 Servlet 程序提供了更为便

利的途径。

另外，JSP 页面和 Servlet 程序是可以相互操作的，也就是说 JSP 页面可以包含从 Servlet 程序输出的内容，可以将内容输出到 Servlet 程序，反过来 Servlet 程序也可以包含从 JSP 页面输出的内容，并且可以将内容输出到 JSP 页面中。JSP 技术的最大优点就是可以将网页的静态内容（HTML 代码）开发与网页的动态内容（Java 代码）开发分隔开来，从而可以使得精通 HTML 但对 Java 不很精通的开发人员专门负责网页静态内容的开发，而那些对 Java 很在行但却不熟悉 HTML 的开发人员就可以专注于网页的动态内容的开发。

第 7 章　数据库的安全性

本章提示

本章共分 5 节，首先介绍了数据库安全性的内涵和数据安全的层次，然后较为详细地阐述了数据库的基本安全机制，包括用户标识和鉴别、存取控制、视图机制、审计等，并以 SQL 为例，讲述了其安全性机制，即视图机构和授权。关于数据库加密，介绍了密码学相关的基本概念、数据库加密的基本要求和特点，以及数据库加密的范围和带来的影响。从用户和密钥两个角度论述了数据库系统的安全性策略，并给出了信息系统的安全级别划分。图 7-1 是本章的知识框图。

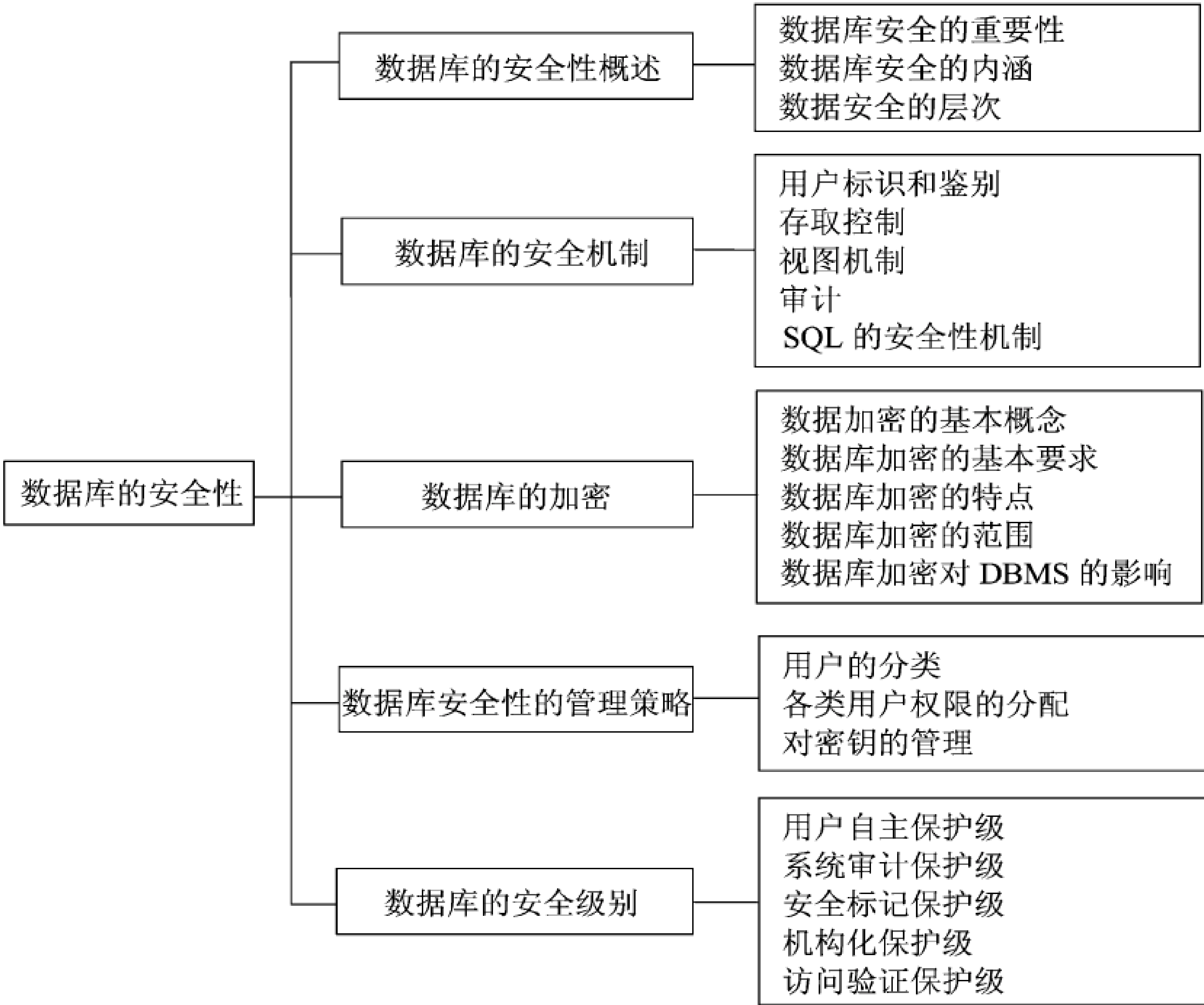


图 7-1　数据库的安全性知识框图

7.1　数据库的安全性概述

数据库的安全性主要是指保护数据库，防止由于非法使用数据库造成数据泄露、更改或破坏。它涉及许多方面的问题，如社会伦理道德、机房安全、口令保密、硬件控制、操

作系统安全、数据库系统等。数据库管理系统提供的主要保护数据安全的手段是对用户存取数据库的数据进行严格的控制。用户存取数据库数据的控制，正常情况下由 DBA 利用数据库管理系统提供的用户管理和授权机制来完成。

数据库数据的共享，必然会带来数据库的安全性问题。数据库中涉及企业、个人的大量数据，其中许多数据可能是非常关键的、机密的或者涉及个人隐私，如果 DBMS 不能严格保证数据库中的数据安全，则会严重制约数据库的应用。

因此，数据库系统中的数据共享不能是无条件地共享，而必须是在 DBMS 统一严格的控制之下，只允许有合法使用权的用户访问系统允许访问的数据。数据库系统的安全保护措施是否有效，是数据库系统的主要性能指标之一。

7.1.1　数据库安全的内涵

数据库受到的各种破坏威胁决定了数据库安全的内涵，数据库的安全是计算机系统安全的重要组成部分。本小节要求介绍数据库受到的破坏威胁、数据库安全的内涵和计算机系统的安全模型。

1．知识点提炼

（1）数据库受到的破坏威胁

① 系统的软件或硬件故障，造成数据被破坏。

② 数据库的并发操作引起数据的不一致。

③ 自然或人为的破坏。

④ 对数据库数据的更新操作有误。

（2）数据库安全的内涵

① 保密性：不允许未经授权的用户存取信息。

② 完整性：只允许被授权的用户修改数据。

③ 可用性：不应拒绝已授权的用户对数据进行存取。

（3）计算机系统的安全模型

在一般计算机系统中，安全措施是一级一级层层设置的。例如可以有如图 7-2 所示的模型。

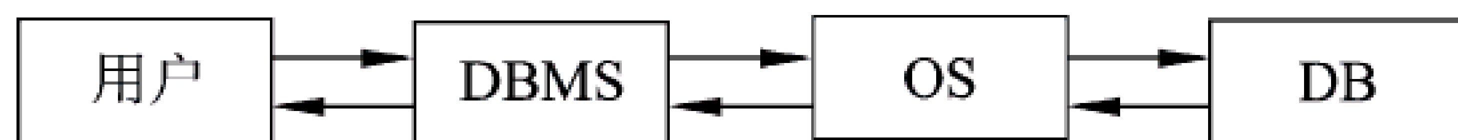


图 7-2　计算机系统的安全模型

2．难点分析

（1）对数据库安全的理解

数据库的安全性是指保护数据库以防止不合法的使用所造成的数据泄露、更改和破坏。安全性问题不是数据库系统所独有的，所有计算机都存在这个问题。只是在数据库系

统中大量数据集中存放，而且为许多最终用户直接共享，从而使安全性问题更为突出。系统安全保护是否有效是数据库系统的一个主要指标。

（2）对计算机安全系统模型的理解

数据库的安全性与计算机系统的安全性是紧密联系、相互支持的，因此在掌握数据库安全之前应对计算机系统安全有一个一般的了解。

在图 7-2 的安全模型中，用户要求进入计算机系统时，系统首先根据输入的用户标识进行身份鉴定，只有合法的用户才准许进入计算机系统。对已进入系统的用户，DBMS 还要进行存取控制。只允许合法用户进行合法操作。操作系统一级也会有自己的保护措施。数据最后还可以以密码形式存储到数据库中。操作系统一级的安全保护措施可以参考操作系统的有关书籍，本章只讨论与数据库有关的用户标识和鉴定、存取控制、视图、密码存储、数据库加密等技术。

3．典型例题

【例 7-1】 判断下列两个说法是否正确，并说明原因。

① 数据库的安全性是指防止数据库被人为地窃取和破坏。

② 为了数据库的安全，重要的数据库应以加密形式存储。

【答案】 ① 不正确　② 正确

【解析】 ① 数据库的安全性不仅指非法用户的人为窃取和破坏，还包括诸如自然的破坏。所以数据库的安全性应包括人为和自然环境两方面的安全。

② DBMS 和操作系统的安全措施并不一定能防止所有非法用户存取数据库，所以数据库以加密形式存储是很有必要的。

7.1.2　数据安全的层次

在信息系统中，数据的安全被划分为 3 个层次：存储安全、管理安全和网络安全。

1．知识点提炼

（1）数据的存储安全

存储设备和存储介质的损坏是数据安全面临的主要威胁之一，自然灾害也是造成数据丢失的重要方面。由于存储设备介质的质量、使用时间等，都可能造成存储设备和介质的失效，导致数据丢失。解决这一问题的主要方法如下。

① 数据冗余技术。

② 分布式存储。

③ 选用质量好的存储设备和介质。

（2）数据的管理安全

企业内部人员操作的失误，人为窃取、破坏是数据安全面临的又一个主要威胁。很难避免操作人员对数据进行误操作，数据被窃取的事情也时有发生。数据的操作失误主要是指用户拥有合法的数据操作权限，主观上不存在恶意的访问，只是因为在操作过程中由于误操作给数据库中的数据造成破坏和泄露。

解决上述问题的方法如下。

① 建立严格的规章制度，并要求操作人员严格遵守和执行。按照操作规程办事，最大限度地减少人为损坏数据的可能性。

② 采用严格的用户权限机制，对所有操作员按其工作性质，授予其数据操作权限。数据库管理系统采用严格的用户认证功能。

（3）数据的网络安全

网络的使用日益普及，而基于网络的数据库系统（如 C/S、B/S 数据库等）的应用更是越来越广泛。由于数据库暴露于公共网络中，不可避免地会遭到来自于网络的攻击。网络信息系统必须建立完善的网络安全策略，防止黑客对数据的窃取、篡改、伪造、破坏。对这一问题的解决方法如下。

① 建立安全的网络环境和安全策略，尽可能减少网络系统的漏洞，防止外来的非法入侵。

② 采用先进的数据加密机制，尽可能地减少明文在网络上的传输和在设备中的存储。

③ 采用有效的认证方法，如数字签名、时间戳等。

2．难点分析

（1）数据库安全 3 个层次之间的关系

数据的安全是一个系统工程，它涉及方方面面的技术和管理问题。在威胁数据安全的因素中，数据存储安全是最基本的，而数据的网络安全是最高层次的。3 个层次相互联系，相互支撑，共同构成数据库的安全体系。

（2）冗余是数据库安全必需的

数据冗余就是将数据备份为多个副本，存储在不同的介质和设备上。当一个设备或介质失效时，可以立即从其他设备或介质中将其恢复，如磁盘镜像等，均属于该技术的实际应用。

（3）分布式存储的概念

将数据存储于网络上的不同计算机中，如果数据损坏或遭到破坏，可以从其他计算机中得到恢复。采用分布式存储的有网络存储（NAS）、IP 存储等。

3．典型例题

【例 7-2】 数据安全的层次从低到高的正确次序是（　　）。

A．管理安全、存储安全和网络安全　　B．存储安全、管理安全和网络安全

C．网络安全、存储安全和管理安全　　D．存储安全、网络安全和管理安全

【答案】 B

7.2 数据库的安全机制

本节介绍了 DBMS 中常用的安全措施，包括用户标识和鉴别、存取控制、视图机制、审计等，并讲述了 SQL 的安全性机制，即视图机构和授权。这些措施可以用来防止由于非

法使用数据库而造成的数据泄露、篡改和破坏。

7.2.1　用户标识与鉴别

1．知识点提炼

任何数据库用户要访问数据库时，都须声明自己的用户标识符。系统首先检查有无该用户标识符的存在。若不存在，自然就拒绝该用户进入系统；但即使存在，系统还要进一步核实该声明者是否确实是具有此用户标识符的用户。只有通过核实的人才能进入系统。这个核实工作就称为用户鉴别。鉴别的方法有以下几种。

（1）口令（password）

口令是使用最为广泛的一种用户鉴别方法。所谓口令，就是注册时 DBMS 给予每个用户的一个字符串。

通过用户名和口令来鉴定用户的方法简单易行，但用户名与口令容易被别人窃取。因此还可以使用更复杂的方法。例如，每个用户都预先约定好一个计算过程或函数，鉴别用户身份时，系统提供一个随机数，用户根据自己预先约定的计算过程或函数进行计算，系统根据用户计算结果是否正确进一步鉴别用户身份。用户可以约定比较简单的计算过程或函数，以便计算起来方便；也可以约定比较复杂的计算过程或函数，以便安全性好。

（2）用户的个人特征

在有更高安全要求的数据库系统中，可以采用根据用户个人特征的方法来鉴别用户。用户的个人特征包括指纹、签名、声波纹等。这些鉴别方法效果不错，但需要特殊的鉴别装置。

（3）磁卡

磁卡是使用较为广泛的鉴别手段，磁卡上记录了某用户的用户标识符。使用时，用户需要显示自己的磁卡，输入设备自动读入用户的用户标识符，然后请求用户输入口令，从而鉴别用户。如果采用智能卡，还可把约定的复杂计算过程存放在磁卡上，结合口令和系统提供的随机数自动计算结果并把结果输入到系统中，安全性更高。

2．难点分析

（1）用户标识的重要性

数据库系统是不允许一个未经授权的用户对数据库进行操作的。用户在访问数据库之前，必须先标识自己的名字和身份，由系统核实，通过鉴定后才提供机器使用特权。也就是说，只有在 DBMS 成功注册了的人员才是该数据库的合法用户，才能使用数据库。因此注册时，每个用户都必须使用自己的且与其他用户不同的用户标识符。

（2）用户标识符与口令的关系

系统在内部存储一个用户标识符和其口令的对应表，用户必须记住自己的口令。当用户声明自己是某用户标识符用户时，DBMS 将进一步要求用户输入口令。系统通过核对口令以鉴别用户的真实身份。核实通过后，系统才确认此用户，才允许该用户进入系统。为

了保密起见，用户在终端上输入的口令是不显示在屏幕上的。

7.2.2　存取控制

数据库安全性所关心的主要是 DBMS 的存取控制机制。数据库安全的最重要一点就是确保只授给有资格的用户访问数据库的权限。对于通过鉴定的合法用户，系统根据其存取权限定义对他的各种操作请求进行控制，确保他只执行合法的操作。这就是存取控制。

1．知识点提炼

（1）存取权限

关系数据库系统中存取控制的数据对象不仅有数据本身，如表、属性列等，还有模式、外模式、内模式等数据字典中的内容，如表 7-1 所示。

表 7-1　关系数据库系统中的存取权限

数据对象		操作类型
模式	模式	建立、修改、检索
	外模式	建立、修改、检索
	内模式	建立、修改、检索
数据	表	查找、插入、修改、删除
	属性列	查找、插入、修改、删除

（2）自主存取控制

任何计算机系统都有两种资源，主动的主体和被动的客体。在数据库系统中，用户、进程等是存取动作的主体；数据、表、记录、元组、字段等是客体。自主存取控制（Discretionary Access Control，DAC）是根据主体身份或主体所属身份或二者的结合，对客体访问进行限制的方法。当主体具有某种访问权并要将该权限授予其他用户时，能自行决定将其访问权直接或间接地转授给其他主体。

DAC 的实现机制是访问控制矩阵，常见的实现方法是访问控制表（ACL）、访问能力表和授权关系表。表 7-2 是一个用户权限定义表的例子。

DAC 在一定程度上实现了多用户环境下的权限隔离和资源保护，易于扩展和理解。但是系统无法控制用户自主授权的情况。要解决这一问题，就需要对系统控制下的所有主客体实施强制存取控制策略。

表 7-2　一个权限定义表的例子

用户名	数据对象名	允许的操作类型	用户名	数据对象名	允许的操作类型
ZHANG	关系 Student	SELECT	WANG	SC.Sno	SELECT
WANG	关系 Student	UPDATE	LIU	关系 Course	INSERT
ZHANG	关系 Course	ALL	WANG	SC.Cno	SELECT
ZHANG	SC.GRADE	UPDATE			

（3）强制存取控制

强制存取控制（Mandatory Access Control，MAC）通过给主体和客体指定安全级，并根据安全级匹配规则来确定某主体是否被准许访问某客体。安全级 L 包括两个元素：密级（Classification）和范围（Categories）。主体的安全级反映的是主体的可信度，客体的安全级则反映客体的敏感度。

当某一客户或某一主体注册进入系统时，系统要求其对任何客体的存取必须遵循以下规则。

① 无上读：主体仅能读取安全级别受此主体安全级别支配的客体的信息；

② 无上写：主体仅能向安全级别受此主体安全级别支配的客体写信息；

③ 无下读：主体仅能读取安全级别支配此主体安全级别支配的客体的信息；

④ 无下写：主体仅能向安全级别支配此主体安全级别的客体写信息。

在实现 MAC 时要首先实现 DAC，DAC 与 MAC 共同构成了 DBMS 的安全机制，如图 7-3 所示。系统首先进行 DAC 检查，对通过 DAC 检查的允许存取的数据对象再由系统自动进行 MAC 检查，只有通过 MAC 检查的数据对象才能被存取。

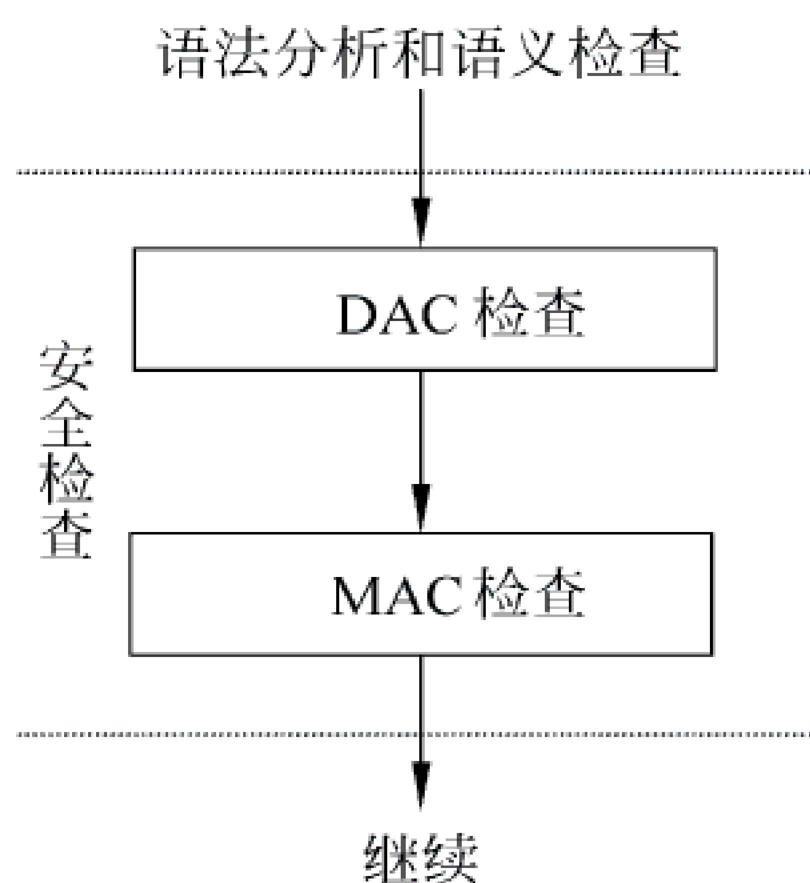


图 7-3　DAC+MAC 安全检查示意图

2．难点分析

（1）存取权限的概念

存取权限是由两个要素组成的：数据对象和操作类型。定义一个用户的存取权限就是要定义这个用户可以在哪些数据对象上进行哪些类型的操作。在数据库系统中，定义存取权限称为授权。这些授权定义经过编译后存放在数据字典中。在关系数据库系统中，DBA 可以把建立、修改基本表的权限授予用户，用户获得此权限后，可以建立和修改基本表、索引、视图等。

（2）对 DAC 的理解

DAC 能够通过授权机制有效地控制其他用户对敏感数据的存取，但无法控制他人将其权限进行非法“扩散”。造成这一问题的主要原因是，DAC 仅仅通过对数据的存取权限来进行安全控制，而数据本身并无安全标记。

（3）对 MAC 的理解

与 DAC 不同，MAC 是对数据本身进行密级标记，无论数据如何复制，标记与数据是一个不可分的整体，只有符合密级标记要求的用户才可以操纵数据，从而提供了更高级别的安全性。MAC 的优点是能够防止特洛伊木马和隐通道的攻击以及防范用户滥用权限。缺点是缺乏灵活性而且强制性太强，使得应用的领域比较窄，一般只用于军事等具有明显

等级的行业或应用领域。

3．典型例题

【例 7-3】 判断下列说法是否正确，并说明原因。

① 在 DBMS 中，每个用户对数据的访问权限是根据实际情况临时赋予的。

② 在数据库系统中，定义存取权限称为授权。这些授权定义经过编译后存放在数据库中。

③ 因为 DAC 控制方式给用户带来了灵活的自主的授权，所以是安全的。

【答案】 ① 不正确　② 不正确　③ 不正确

【解析】

① 在数据库系统中，为了保证用户只能访问其有权存取的数据，必须预先对每个用户定义存取权限。

② 授权定义经过编译后存放在数据字典中。在用户发出存取数据库操作的请求后，DBMS 查找数据字典，根据其存取权限对操作的合法性进行检查，若用户的操作请求超出了定义的权限，系统将拒绝执行该操作。

③ DAC 的优点是简单、灵活，但是由于用户对数据的存取权限是“自主”的，用户可以自由地决定将数据的存取权限授予任何人，决定是否也将“授权”的权限授予别人，而系统对此无法控制。在这种授权机制下，仍可能存在数据的“无意泄露”。

7.2.3　视图机制

通过视图机制把要保密的数据对无权存取的用户隐藏起来，从而自动地对数据提供一定程度的安全保护。

1．知识点提炼

视图的概念。

进行存取权限控制时，可以为不同的用户定义不同的视图，把数据对象限制在一定的范围内，即把用户可以使用的数据定义在视图中，这样用户就不能使用超出视图定义范围的其他数据，从而保证了数据库的安全性。

例如，每个车间的统计员只能查询本车间职工的情况，可为他们分别定义只包括本车间职工记录的视图；为保密职工工资情况，可定义一个不包含工资属性的视图，供一般查询使用。

2．难点分析

掌握视图的本质特征。

视图是取自数据库一部分内容建立起来的表，但视图仅仅是一个定义，并不是实际存在的一个数据库，因此，视图本身没有数据，不占用磁盘存储空间。视图的使用使系统具有 3 个优点：数据安全性、数据独立性和操作简便性。

7.2.4　审计

审计功能是 DBMS 安全性方面的重要组成部分。任何系统的安全保护措施都不是无懈可击的，蓄意盗窃、破坏数据的人总是想方设法打破控制。因此安全工作者除了要使系统在一定代价内尽量可靠外，还必须考虑能发现越权或企图越权的行为，这就是审计的目的。

1．知识点提炼

（1）审计的概念

审计仅是一种监视措施，它把用户对数据的所有操作都自动记录下来，放入审计日志。事后，DBA 可利用审计日志中的记录，重现导致数据库现有状况的一系列事件，找出非法存取数据的人、时间和内容等，以便追查有关责任；同时审计也有助于发现系统安全方面的弱点和漏洞，在未产生问题时分析有无潜在的问题。

（2）审计的内容

审计内容一般包括本次操作的有关值，如操作类型、操作者、操作时间、数据对象、操作前值和操作后值等，但也可以增加一些内容。例如，可以在用户的磁卡中增加一个数据项——用户访问数据库的次数，在系统中记录用户访问本系统的次数，且分别自动增加。一旦某用户的这两个数据不一致，就可抓住有潜在问题的地方。

2．难点分析

认识审计存在的问题及使用方式。

审计常常花费很大的代价。例如，对于粒度过细（记录每个元组值的改变）的审计日志，是很费时间和空间的。特别是在大型分布和数据复制环境下的大批量、短事务处理的应用系统中，实际上是很难实现的。所以，审计机制一般只在有较高安全要求的场合被采用。在实际 DBMS 中，审计往往是一个可选项，由 DBA 决定是否采用、何时采用。

7.2.5　SQL 语言中的安全性控制

在了解了上述数据库基本安全机制的基础上，就可以考察 SQL 语言的安全性控制措施。SQL 有两个功能用于安全性机制：一个是视图机构，它可以用来对无权访问数据的用户进行数据屏蔽；二是授权子系统本身，它允许有特定存取权的用户有选择地和动态地把各种数据访问权限授予相应的用户。

1．知识点提炼

（1）SQL 的视图（VIEW）

在 SQL 语言中，创建视图的语句及格式如下：

CREATE VIEW〈视图名〉[〈列名清单〉]
　　AS〈子查询〉
　　[WITH CHECK OPTION]

这里，〈子查询〉为任一合法的 SELECT 语句（但一般不含有 ORDER BY、UNION

等语法成分）；WITH CHECK OPTION 是一个可选项，当有它时，表示今后对此视图进行 INSERT、UPDATE 和 DELETE 操作时，系统会自动检查视图是否符合原定义视图时子查询中的〈条件表达式〉。不符合时，就不执行。

在 SQL 语言中，可以把给定的用户限制在关系的垂直方向的子集上，或者水平方向的子集上（见例 7-4 中的①、②和③），或者统计的总和上（见例 7-4 的④）。

（2）SQL 的授权（GRANT）

在 SQL 语言中，授权其他用户使用表和视图的语句为 GRANT，它的格式如下：

GRANT〈特权清单〉ON〈表名或视图名〉TO〈用户名〉

[WITH GRANT OPTION]

这里提及的特权主要有以下 8 种。

① 对表或视图的操作

SELECT

UPDATE

INSERT

DELETE

② 对表建立索引

INDEX

③ 对表追加字段

EXPAND

④ 授予以上全部的 6 个特权

ALL

⑤ 授予 DBA 特权

DBA

一般来说，当系统初始安装时，系统必须至少识别一个用户具有专门特权 DBA。DBA 特权允许持有者能够完成系统中任何有效的操作。此外，当授予某用户特权后，该用户无权将这些特权传递给其他用户。为了使用户能够传递授权，可以使用子句：WITH GRANT OPTION。

（3）收回权限（REVOKE）

当用户将某些权限授予其他用户后，还可以使用 REVOKE 语句将权限收回。REVOKE 语句的格式如下：

REVOKE〈特权清单〉[ON〈表名和视图名〉]

FROM〈用户名〉

CASCADE

说明：只有使用 GRANT 语句授出了权限的用户才能使用本语句收回自己授出去的权限。选项 CASCADE 表示，如果〈用户名〉将其权限又转授给了其他用户，那么这些权限

也将从其他用户处收回。

2．典型例题

【例 7-4】 视图的建立。某关于学生选课的数据库中各表的模式定义如下。

学生关系模式：S（S#，SNAME，AGE，SEX）

学习关系模式：SC（S#，C#，GRADE）

课程关系模式：C（C#，CNAME，TEACHER）

写出建立以下视图的正确 SQL 语句。

① 对于允许读取学生成绩 S（S#，SNAME，CNAME，GRADE）的用户定义视图。

② 对于只允许读取关系 SC 中及格成绩的用户定义视图。

③ 对于只允许读取自己成绩的用户定义视图。

④ 对于只允许读取每个学生的平均成绩，而不允许读取个别成绩的用户定义视图。

【答案】

```
① CREATE VIEW V1
   AS SELECT S, S#, SNAME, CNAME, GRADE
      FROM S, SC, C
      WHERE S.S# = SC.S# AND SC.C# = C.C#
② CREATE VIEW V2
   AS SELECT *
      FROM SC
      WHERE GRADE>=60
③ CREATE VIEW V3
   AS SELECT *
      FROM SC
      WHERE S# = (SELECT S#
                  FROM S
                  WHERE SNAME = user)
```

这里，user 是 SQL 的保留字，表示当前用户。

```
④ CREATE VIEW V4
    AS SELECT S#, AVG(GRADE)
       FROM SC
       GROUP BY S#
```

【例 7-5】 写出实现以下功能的 SQL 语句。

① 使用户 ZHANG 不仅获得查询和修改表格 S 的权限，而且 ZHANG 还可以把这个权限转授给其他用户。

② 将表 S 的 INSERT 权限从用户 ZHANG 处收回，并收回 ZHANG 转授的该权限。

【答案】

```
① GRANT SELECT, UPDATE ON S TO ZHANG
   WITH GRANT OPTION
② REVOKE INSERT ON S
   FROM ZHANG
   CASCADE
```

7.3　数据库的加密

数据是存储在介质上的，数据还经常通过通信线路进行传输。对于特别重要和高度敏感的数据，例如财务数据、军事数据、国家机密等，除上一节所提及的安全措施外，必须考虑到各种非法存取或破坏数据的可能性。在这种情况下，还可以采用数据加密技术，以密码形式存储和传输数据。这样，即使非法存取者进入了系统，窃取了数据，没有密钥也不能对数据进行解密。因此，数据加密是防止数据库中数据在存储和传输中失密的有效手段。

7.3.1　数据加密的一些基本概念

1．知识点提炼

加密的基本思想是根据一定的加密算法将原始数据变换为不可直接识别的密文，从而使得不知道解密算法的人无法获知数据的内容。

未加密的信息称为明文（Plain Text），用加密算法和密钥将明文加密后得到的信息称为密文（Cipher Text）。由明文到密文的变换称为加密，合法接收者从密文恢复出明文的过程称为解密（或脱密），非法接收者试图从密文分析出明文的过程称为破译。

对明文进行加密时采用的一组规则称为加密算法。对密文解密时采用的一组规则称为解密算法。加密算法和解密算法是在一组仅有合法用户才知道的秘密信息（称为密钥）的控制下进行操作，加密和解密过程使用的密钥分别称为加密密钥和解密密钥。

明文记为 P 且 P 为字符序列， P=[P1,P2,…,Pn]，密文记为 C，C=[C1,C2,…,Cn]，明文到密文之间的变换记为：

$C=E_{K1}(P)$

其中 E 为加密算法，K1 为加密密钥。密文到明文的变换记为：

$P=D_{K2}(C)$

其中 D 为解密算法，K2 为解密密钥。要求密码系统满足：$P=D_{K2}(E_{K1}(P))$。加密和解密过程如图 7-4 所示。

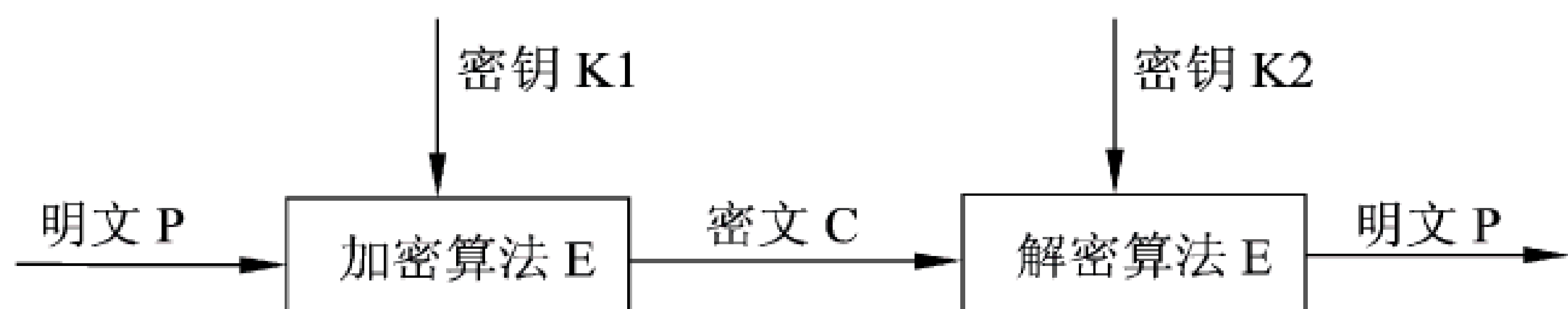


图 7-4　数据的加密和解密过程

根据加密密钥和解密密钥是否相同，将加密算法分为对称密钥算法（Symmetric Cipher）和非对称密钥算法（Asymmetric Cipher）。若加密密钥和解密密钥相同，或实质上等同，即从一个易于推出另一个，则称加密算法为对称密钥算法；若加密密钥和解密密钥不相同，且从一个很难推出另一个，则称加密算法为非对称密钥算法，又称公开密钥算法。公开密钥算法用一个密钥进行加密，而用另一个密钥进行解密，其中的加密密钥可以公开，又称为公开密钥（Public Key），简称公钥。解密密钥必须保密，又称为私人密钥（Private Key），简称私钥。

2．典型例题

【例 7-6】 以下各题可供选择的答案中只有一个是正确的，请选择正确的答案填入括号中。

（1）一般说来，对数据进行加密时，（　　）。

A．仅使用加密算法，不需要密钥　　B．仅使用密钥，不需要加密算法

C．既要使用加密算法，也要使用密钥　　D．或者使用加密算法，或者使用密钥

（2）在非对称密钥算法中，需要加密时可以使用（　　）。

A．公开密钥　　B．私人密钥

C．公开密钥和私人密钥　　D．公开密钥或私人密钥

【答案】（1）C　（2）D

【解析】

（1）对明文进行加密时，加密算法和密钥缺一不可。加密算法是指对明文进行加密时采用的规则，而密钥则是加密过程中使用的秘密信息。通常，加密的算法可以公开，但密钥是要保密的，尤其是在对称密钥体制中。

（2）在非对称密钥算法中，用户拥有一对密钥，其中一个密钥是公开的，简称公钥，另一个是私有的，简称私钥。加密时既可以使用公钥，也可以使用私钥。用公钥加密的密文，一定用私钥才能解密，反之亦然。

7.3.2　数据库加密的基本要求和特点

1．知识点提炼

（1）数据库加密的基本要求

一个良好的数据库加密系统应该满足以下基本要求。

① 字段加密

传统的加密以报文为单位，加密从头至尾顺序进行。数据库数据的使用方法，决定了不可能以整个数据库文件为单位进行加密。

② 密钥动态管理

数据库客体之间隐含着复杂的逻辑关系，一个逻辑结构可能对应着多个数据库物理客体，所以数据库加密不仅密钥数量大，而且组织和存储工作十分复杂，需要对密钥实现动

态管理。

③ 合理处理数据

数据的合理处理包括几方面的内容。首先要恰当地处理数据类型，否则 DBMS 将会因加密后的数据不符合定义的数据类型而拒绝加载；其次，需要处理数据的存储问题，实现数据库加密后，应基本上不增加空间开销。

④ 不影响用户的合法操作

加密系统影响数据操作的响应时间应尽量短。此外，对数据库的合法用户来说，数据的录入、修改和检索操作应该是透明的，不需要考虑数据的加密和解密问题。

（2）数据库加密的特点

较之传统的数据加密技术，数据库密码系统有其自身的要求和特点。

① 数据库密码系统应采用公开密钥

数据库数据是共享的，有权限的用户需要知道密钥来查询数据。因此，数据库密码系统宜采用公开密钥的加密方法。

② 多级密钥结构

数据库关系运算中参与运算的最小单位是记录的字段，查询路径依次是库名、表名、记录名和字段名。因此，字段是最小的加密单位。也就是说，当查得一个数据后，该数据所在的库名、表名、记录名、字段名都应是知道的。对应的库名、表名、记录名、字段名都应该具有自己的子密钥，这些子密钥组成了一个能够随时加密和解密的公开密钥。

③ 数据库的加密机制

公开密钥的加密体制不适合于数据库加密。数据库加密密钥和解密密钥应该是相同、公开的，而加密算法应该是绝对保密的。

数据库公开密钥加密机制应是一个二元函数：

密文=F(密钥，明文)

当加密算法 F 确定之后，只要给出密钥和待加密的明文，即可得到相应的密文。解密过程即是加密过程的逆过程：

明文=F^{-1}(密钥，密文)

由此可知，数据库密码的加密机制应是既可加密又可解密的可逆过程。

④ 加密算法

加密算法是数据加密的核心，一个好的加密算法产生的密文应该频率平衡，随机无重码规律，周期很长而又不可能产生重复现象。窃密者很难通过对密文频率、重码等特征的分析获得成功。同时，算法必须适应数据库系统的特性，加密和解密响应迅速。

著名的 MH 背包算法就是一种适合数据库加密的算法。它的基本思想是：有一个函数 F，使 X=F(K,Y)，在这里 X 相当于公开密钥向量，Y 相当于明文。在算法 F 不公开的情况下，若已知 K 和 X，要还原出 Y 来，穷尽次数为 2^n 次。如果向量的分量 n 较大时，用穷尽的方法来还原将是十分困难的。

2．典型例题

【例 7-7】 判断下列说法是否正确，并说明原因。

① 数据库的加密是以整个文件为单位的。

② 数据库密码系统采用公开密钥主要基于安全的考虑。

③ 数据库的加密应采取密钥公开、算法保密的机制。

【答案】 ① 不正确　② 不正确　③ 正确

【解析】

① 一般来说，加密和解密的粒度应是每个记录的字段数据。如果以文件或列为单位进行加密，必然会形成密钥的反复使用，从而降低加密系统的可靠性或者因为加密和解密时间过长而无法使用。只有以记录的字段数据为单位进行加密和解密，才能适应数据库操作，同时进行有效的密钥管理并完成“一次一密”的密码操作。

② 在传统的密码系统中，密钥是秘密的，相对而言，知道的人越少越好。一旦获取了密钥和密码体制就能解开密文。而数据库数据是要共享的，有权限的用户随时需要知道密钥来查询数据。设想数据库密码系统的加密算法是保密的，而且具有相当的强度，那么利用密钥，采用 OS 和 DBMS 层的工具，也无法得到数据明文。所以数据库密码系统采用公开密钥主要是基于使用方便的考虑。

③ 有些公开密钥体制的密码，如 RSA 密码，其加密密钥是公开的，算法也是公开的，但是其算法可能因人而异。作为数据库密码的加密算法不可能因人而异，因为寻找这种算法有其自身的困难和局限性，机器中也不可能存放很多种算法，因此典型的公开密钥的加密体制并不适合于数据库加密。数据库加密密钥和解密钥应该是相同、公开的，而加密算法应该是绝对保密的。

7.3.3　数据库加密的范围和影响

数据加密通过对明文进行复杂的加密操作，以达到无法发现明文和密文之间、密文和密钥之间的内在关系，也就是说经过加密的数据经得起来自 OS 和 DBMS 的攻击。另一方面，DBMS 要完成对数据库文件的管理和使用，必须具有能够识别部分数据的条件。据此，只能对数据库中数据进行部分加密。此外，数据的加密对原数据库系统的功能会产生一定的影响。

1．知识点提炼

（1）数据库加密的范围

① 索引字段不能加密

为了达到迅速查询的目的，数据库文件需要建立一些索引。不论是字典式的单词索引、B 树索引或 HASH 函数索引等，它们的建立和应用必须是明文状态，否则将失去索引的作用。有的 DBMS 中可以建立簇聚索引，这类索引也需要在明文状态下建立和维护使用。

② 关系运算的比较字段不能加密

DBMS 要组织和完成关系运算，参加并、差、积、商、投影、选择和连接等操作的数据一般都要经过条件筛选，这种“条件”选择项必须是明文，否则 DBMS 将无法进行比较筛选。例如，要求检索工资在 1000 元以上的职工人员名单，“工资”字段中的数据若加密，SQL 语句就无法辨认比较。

③ 表间的连接码字段不能加密

数据模型规范化以后，数据库表之间存在着密切的联系，这种相关性往往是通过“外部编码”联系的，这些编码若加密就无法进行表与表之间的连接运算。

（2）数据库加密对 DBMS 的影响

目前 DBMS 的功能比较完备，特别像 Oracle、Sybase 这些采用 Client/Server 结构的数据库管理系统，具有数据库管理和应用开发等工具。然而，数据库的数据加密以后，DBMS 的一些功能将无法使用。

① 无法实现对数据制约因素的定义

有的数据库系统（如 Sybase）的规则定义了数据之间的制约因素。数据一旦加密，DBMS 将无法实现这一功能，而且值域的定义也无法进行。此外，加密后的数据仍然受到原来字段长度的制约。

② 密文数据的排序、分组和分类

SELECT 语句中的 GROUP BY、ORDER BY、HAVING 子句分别完成分组、排序、分类等操作。这些子句的操作对象如果是加密数据，那么解密后的明文数据将失去原语句的分组、排序、分类作用，显然这不是用户所需要的。

③ SQL 语言中的内部函数将对加密数据失去作用

DBMS 对各种类型数据均提供了一些内部函数，这些函数不能直接作用于加密数据。

④ DBMS 的一些应用开发工具的使用受到限制

DBMS 的一些应用开发工具不能直接对加密的数据进行操作，因而它们的使用也会受到一定的限制。

2．典型例题

【例 7-8】 判断下列说法是否正确，并说明原因。

① 索引字段不能加密的原因是因为它不是可加密的数据类型。

② 数据库数据被加密后，其长度不受原来字段长度的制约。

③ DBMS 定义的函数不能直接作用于加密的数据。

【答案】 ① 不正确 ② 不正确 ③ 正确

【解析】

① 索引字段的建立和使用必须在明文的状态下进行，如果对其进行加密，则其索引功能就会失效。所以，索引字段不能加密的原因不是因为它不是可加密的数据类型，而是基于使用的角度来考虑的。

② 数据库中的每个字段的类型、长度都有具体的限定。数据加密时，数值类型的数据只能在数值范围内加密，日期和字符类型的数据也都只能在各自的类型范围内加密，密文长度也不能超过字段限定的长度，否则 DBMS 将无法接受这些加密过的数据。

③ 由于 DBMS 定义的函数不能直接识别加密的数据，因此它作用于加密的数据没有任何意义。

7.4　数据库安全性的管理策略

7.4.1　对用户的管理

在 DBMS 系统中，通常将用户分为以下几个级别。

① 系统管理员：负责整个系统的安全，其权限最高，管理整个网络资源以及域服务器资源、文件服务器资源等。

② 网络管理员：负责网络资源的安全，管理网络设备、广域网访问权限控制等。

③ 数据库管理员：负责数据库的安全，主要管理数据库资源。

④ 开发人员：在安全授权下使用各种资源，可对本地工作站及开发所涉及的数据、文件、设备等拥有最高的权限。

⑤ 高级用户：在安全授权下使用各种资源，能使用网络资源，对本地工作站拥有一定的权限，可使用本地的一部分资源（如使用硬盘，执行硬盘上已安装的软件等）。

⑥ 一般用户：在安全授权下使用指定的资源。对本地工作站只拥有执行指定程序的权限，不可直接对任何资源进行操作，只能通过指定的程序对指定的资源进行操作。

通过对上述人员的合理管理实现对资源的有效控制，杜绝对数据的无权访问。

7.4.2　对密钥的管理

密钥的管理是复杂的，而且对于加密的数据库的安全起了决定性的作用。密钥的管理包括生成密钥值所需的特征，要让需要使用密钥的特定系统事先知道密钥，要保护密钥不被泄露或替换。密钥管理方法主要包括以下几个方面。

① 密钥的建立：包括生成密钥和发布密钥。这主要取决于使用的加密算法和密钥协议。

② 密钥的备份和恢复：密钥的备份和恢复是指密钥丢失或由于其他原因无法获得后，恢复密钥的备份。要注意的是，如果信息是以加密的形式存储的并且需要用特定的密钥来解密的话，那么密钥的丢失可能就意味着信息的丢失。

③ 密钥的替换和更新：密钥的替换和更新是指在密钥使用期满或在特殊情况下重新建立密钥的过程。一般来说，所有的密钥都有使用期限，一是因为密钥有可能受到攻击者的分析，密钥使用的时间越长，破译的可能性就越大；二是因为密钥有可能被无意泄露。

④ 密钥的吊销：密钥的吊销是指在特殊情况下停止密钥的使用。

⑤ 密钥的期满和终止：在密钥期满后要停止该密钥的使用，这会涉及密钥的销毁和归档。密钥销毁是指删除所有与某个密钥有关的信息。密钥还需要进行归档保存，尤其是在非对称加密算法中，密钥可能要用于数字签名等。

7.5　数据库的安全级别

我国数据库安全国家标准，即计算机信息系统安全等级划分准则（GB17859-1999）在 1999 年正式颁布，该标准把安全级别划分为五级。其中 B3 级为最高级，C1 级为初始级。

（1）第 1 级：用户自主保护级（C1 级）

满足该级别的系统具有如下功能：主体、客体及主客体分离，身份标识与鉴别；自主访问控制。其安全核心是自主访问控制。

（2）第 2 级：系统审计保护级（C2 级）

在 C1 级标准基础上，增加审计功能。这一级的安全核心是审计。

（3）第 3 级：安全标记保护级（B1 级）

在 C2 级标准基础上，增加强制访问控制功能。这一级的安全核心是强制访问控制。

（4）第 4 级：机构化保护级（B2 级）

在 B1 级标准基础上，增加隐蔽通道和数据库安全的形式化功能。这一级的安全核心是隐蔽通道和形式化。

（5）第 5 级：访问验证保护级（B3 级）

在 B2 级标准基础上，增加访问监控器。这一级的安全核心是访问监控器。

根据我国对计算机信息系统安全保护划分准则，只要合理设计数据库系统，就可以达到 C2 级。虽然 C2 级标准数据安全有一定的保障，但在网络环境下还是不够的。为了防止网络用户的非法访问，还须对访问数据的权限采取更严格的控制，如强制访问控制。这样的数据库系统可达到 B1 级。

当前，大部分的数据库管理系统可达到 B1 级，B1 级表示数据库的主体、客体分离（数据独立），具有身份鉴别、自主访问控制，以及审计功能。只有达到了 B1 级的安全标准，才能有效地防止网络上用户的恶意入侵和合法用户对权限的滥用，保证数据库真正的安全。

练习题

选择题（以下各题中只有一个正确答案，请将选择写入括号中）

1．数据库安全的内涵不包括以下哪一项？（　　）

A．完整性　　B．合法性　　C．可用性　　D．保密性

2．计算机系统的安全模型由哪几部分组成？（　）

A．用户、数据库管理系统、操作系统、数据库

B．用户、数据库管理系统、操作系统、加密系统、数据库

C．数据库管理系统、操作系统、加密系统、数据库

D．数据库管理系统、操作系统、网络系统、数据库

3．在威胁数据安全的因素中，（　）是最基本的。

A．管理安全　B．网络安全　C．传输安全　D．存储安全

4．当一个用户要进入数据库系统时，系统首先检查有无该用户标识符的存在，这个工作称为（　）。

A．用户鉴别　B．用户认证　C．用户标识　D．用户登录

5．组成数据存取权限的两个要素是（　）。

A．用户标识符和口令　B．数据对象和级别

C．数据对象和操作类型　D．用户级别和数据对象

6．以下哪种资源不属于客体？（　）

A．字段　B．表　C．进程　D．属性

7．在强制存取控制中，主体仅能向安全级别受此安全级别支配的客体写信息的规则称为（　）。

A．无上写　B．无下写　C．无上读　D．无下读

8．一般来讲，DBMS 的安全机制是怎样构成的？（　）

A．只进行 MAC 检查人员

B．只进行 DAC 检查

C．先进行 MAC 检查，再进行 DAC 检查

D．先进行 DAC 检查，再进行 MAC 检查

9．关于视图，以下哪种说法是正确的？（　）

A．视图既包括数据，也占用磁盘空间

B．视图不包括数据，但要占用磁盘空间

C．视图既不包括数据，也不占用磁盘空间

D．视图不包括数据，但要占用磁盘空间

10．审计往往是一个可选项，由（　）决定是否采用和何时采用。

A．DBMS　B．DBA　C．用户　D．网管

11．SQL 语言有两个功能用于安全性机制，它们是（　）。

A．用户标识符和口令　B．授权和操作

C．视图和授权　D．视图和口令

12．设数据库中有模式 S、SC 和 C，（如 7.2.5 中所示），建立选修课程号为 B2 的学生的视图，正确的命令是（　）。

A．CREATE VIEW COMPUTER（S#，SNAME，GRADE）
　　AS SELECT S#，SNAME，GRADE
　　FROM S，SC
　　WHERE S.S#= SC.S# AND SC.C#='B2'

B．CREATE VIEW COMPUTER（S.S#，S.SNAME，SC.GRADE）
　　AS SELECT S#，SNAME，GRADE
　　FROM S，SC
　　WHERE S.S#= SC.S# AND SC.C#='B2'

C．CREATE VIEW COMPUTER（S.S#，S.SNAME，SC.GRADE）
　　AS SELECT S.S#，S.SNAME，SC.GRADE
　　FROM S，SC
　　WHERE S.S#= SC.S# AND SC.C#='B2'

D．CREATE VIEW COMPUTER（S#，SNAME，GRADE）
　　AS SELECT S.S#，S.SNAME，SC.GRADE
　　FROM S，SC
　　WHERE S.S#= SC.S# AND SC.C#='B2'

13．基本表 S 的属主把查询表 S 的权限授予用户 U1，同时授予 U1 可以传递授权，正确的命令是（　　）。

A．GRANT SELECT ON S
　TO U1
　WITH GRANT OPTION

B．GRANT SELECT TO S
　ON U1
　WITH GRANT OPTION

C．GRANT SELECT ON S
　TO U1
　WITH GRANT

D．GRANT SELECT TO S
　ON U1
　WITH GRANT

14．数据库的加密应以（　　）为单位。

A．字段　　B．表　　C．数据库　　D．属性

15．数据库密码系统应采用（　　）密码体制。

A．对称密钥　　B．公开密钥　　C．私有密钥　　D．加密密钥

16．一般来讲，数据库中的（　　）字段不能加密。

A．数值　B．日期　C．字符　D．索引

17．在 DBMS 中，（　）具有最高的权限。

A．网络管理员　B．数据库管理员　C．系统管理员　D．高级用户

18．在 DBMS 中，（　）负责数据库的安全，主要管理数据库资源。

A．网络管理员　B．数据库管理员　C．系统管理员　D．高级用户

19．在计算机信息系统安全等级划分准则中，（　）级的安全性最低。

A．C1　B．C2　C．B1　D．B2

20．目前，大部分的数据库管理系统的安全性可以达到（　）级。

A．C1　B．C2　C．B1　D．B2

练习题答案

1．B　2．A　3．D　4．A　5．C　6．B　7．A　8．D
9．C　10．B　11．C　12．D　13．A　14．A　15．B　16．D
17．C　18．B　19．A　20．C

第 8 章　数据库发展趋势与新技术

本章提示

传统的层次、网状和关系数据库系统在许多传统的商业数据库应用中取得了极大的成功，然而在设计和实现更为复杂的数据库应用时，传统数据库系统就暴露了一些缺陷。在设计与实现工程设计和制造数据库、科学实验数据库、电信数据库、地理信息系统数据库以及多媒体数据库的时候，新的应用要求被提出了，如长事务的处理，图像或大文本项等新数据类型的存储，以及非标准的特殊应用操作，传统的数据库系统往往不能满足这些复杂数据库应用的要求。为了适应不断发展的信息化建设的需要，很多结合原有理论同时又有所创新的数据库理论、标准、以及商用产品逐渐被推广应用。本章将根据考试大纲的要求，主要是讲其中的两个逐渐成熟的且已大规模商业化运行的数据库理论和标准。也介绍了国内目前比较关注的 ERP 系统，将数据库技术与 ERP 的设计、实施、运行相结合，力求对从事和关心此项应用的读者有所帮助和启迪。同时对各项环节的重点和难点做出详略恰当的提炼。在详细的典型例题分析之后，还将给出适量的实战练习题，以加深对这些知识的理解。本章共分 3 节，如图 8-1 所示的是本章的知识框图。

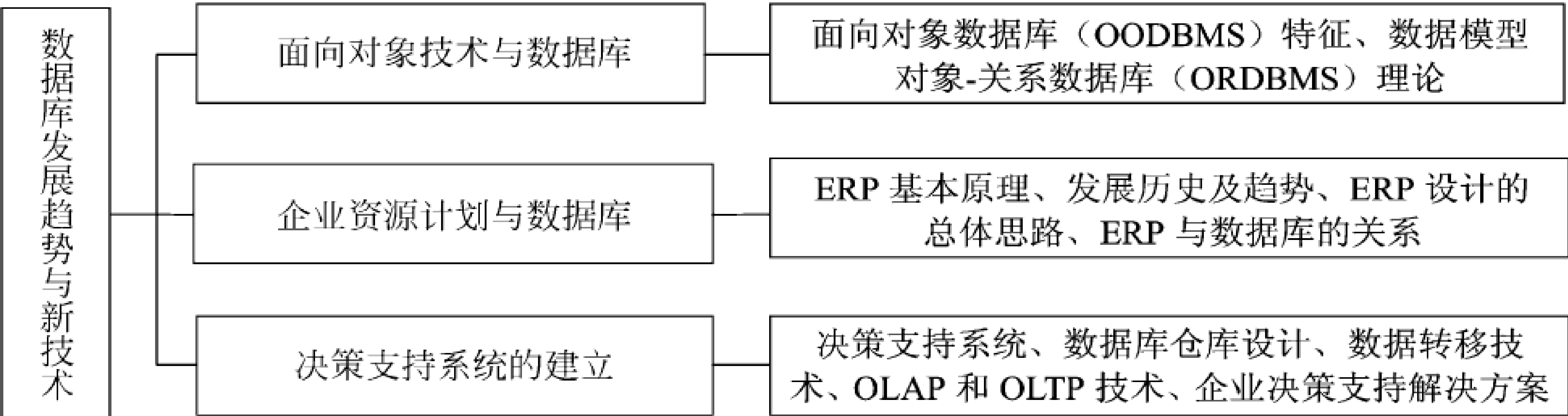


图 8-1　数据库发展趋势与新技术知识框图

8.1　面向对象数据库

数据库技术与面向对象程序设计方法相结合形成了面向对象数据库系统（Object Oriented Database System，OODBS），对象-关系数据库系统（Object-Relationship Database System，ORDBS）。

面向对象程序设计方法已经被广泛的应用于软件工程、知识库、人工智能和计算机系统等领域。面向对象程序设计方法和数据库技术的结合，不但能让设计者定义复杂对象的结构，还能让设计者定义作用于这些复杂对象的操作，从而能够有效地支持新一代的数据

库应用。

面向对象数据库产品的研制和开发上存在着两大派别，即对象-关系数据库和纯粹的面向对象数据库。前者认为关系数据库具有坚实而成熟的理论基础，主张对现有的关系数据库系统进行扩充和改进，使之升级为对象关系数据库系统。具有代表性的对象关系数据库系统产品有：DB2、Smalltalk/SQL、Dbkit、CommonBase、Oracle 10、SQL Server 2000 等。纯粹的面向对象数据库派则主张进行彻底的数据库革命，即采用全新的数据模型和模式，抛开现有的数据库系统，从底层做起，使之成为真正的、纯粹的面向对象数据库系统。其代表性的产品有 Objectstore、ONTOS、Versant、IRIS 及 Orion 等。无论是对象-关系数据库还是纯粹的面向对象数据库，面向对象的概念和方法是其不可缺少的组成部分。究竟哪一个更适合于存储和访问复杂的数据，具有更优越性的性能，在理论界和工业界还有争论，有待于在实际应用中加以比较和校验。图 8-2 根据大纲要求给出面向对象技术与数据库的知识框图。

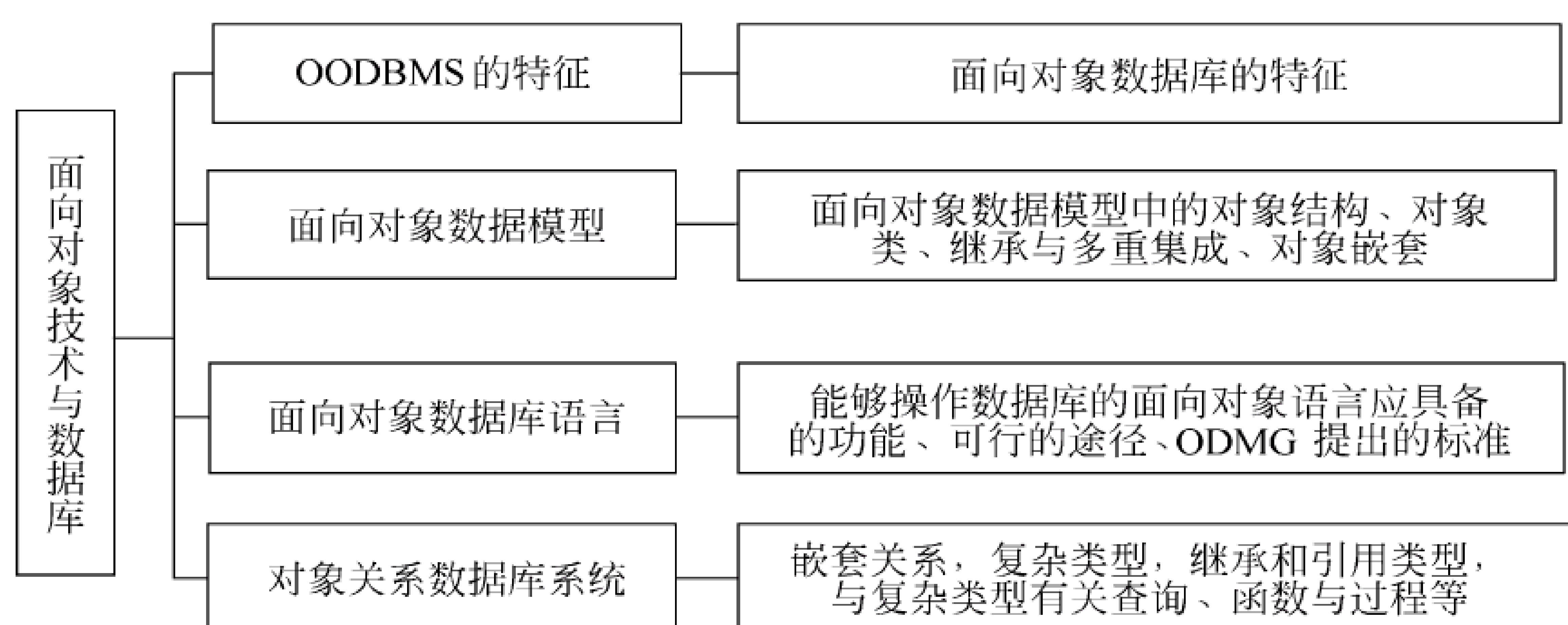


图 8-2　面向对象技术与数据库知识框图

8.1.1　面向对象数据库系统的特征

1．知识点提炼

数据库的特征依赖于实际应用，所设计的数据库语言必须允许用户方便地使用这些特征，数据库的结构也应能有效地支持这些特征。本节结合面向对象的程序设计方法，讨论面向对象数据库系统与传统数据库系统相区别的主要特征。

① 面向对象数据库系统应该具有表达和管理对象的能力。面向对象数据库系统通过对象及它们之间的相互联系来描述现实世界。它应该支持对象标识，使得对象的存在不依赖于本身的值，而只依赖于它的标识，对象间能够通过对象标识而相互区分。类的层次和继承是一个关键的概念，新的类允许从以前定义过的类那里继承结构和操作。因此在面向对象数据库系统中，新的对象类型可以简便地重用已有的类型定义。面向对象数据库系统的一个问题是如何表示对象间的联系。在 ODMG 2.0（Object Database Management Group）即对象数据库管理组标准中，提出了用一对反向引用来表示二元关系，就是把与某个对象

相关的对象的标识放在那个对象内部，并维护参照完整性。

② 面向对象数据库系统中的对象可以具有任意复杂度的对象结构。这使对象能够包含所有描述该对象的必要信息。传统数据库恰好与此相反，它把关于复杂对象的信息分散在许多关系或记录中，从而丧失了现实世界的对象与数据库表示之间的直接对应关系。在面向对象数据库系统中，允许逐步细化复杂实体，还能将整个复杂对象或其子集作为一个独立的单位，可以在某一时刻将一成员对象加进去。

③ 面向对象数据库系统必须具有与面向对象编程语言交互的接口。面向对象程序设计中的对象是瞬态对象，只在程序执行过程中存在，而面向对象数据库可以延长对象的存在，把对象持久地存储起来。对象在程序结束之后仍然持续存在，以后可以被检索或被其他程序使用。面向对象数据库系统通过与面向对象编程语言交互的接口，可以提供持久化对象和共享对象的能力，从而允许多个程序和应用共享这些对象。

④ 面向对象数据库应具有表达和管理数据库变化的能力。管理同一对象的多个版本的能力对于设计和工程应用是至关重要的。一个对象的旧版本代表一个已经通过测试和鉴定的设计方案，那么应该保存这个版本直到新版本通过测试和鉴定。除了允许版本变化外，面向对象数据库系统也应该允许模式演变。所谓模式演变，是指类的声明发生了变化，或创建了新的类或联系。

综上所述，一个面向对象数据库系统首先应是一个数据库系统，同时又必须具有面向对象的特征。

2．典型例题

【例题 8-1】 讨论的面向对象数据库系统的主要特征，主要是与传统数据库系统相区别的，在面向对象数据库系统中，要求面向对象数据库系统应该具有表达和管理对象的能力，也就是说它应该支持__①__，而不依赖于对象本身的值，对象间只需要通过__①__就能够相互区分。在面向对象数据库系统与面向对象编程语言交互的接口，可以提供__②__和共享对象的能力，从而允许多个程序和应用在整个数据库操作周期内访问和共享这些对象。

① A．类名　　B．数据类型定义　　C．值域标识　　D．对象标识

② A．数值定义　　B．持久化对象　　C．定义对象　　D．删除对象

【答案】 ① D　② B

【解析】 作为面向对象的数据库系统应该和面向对象的语言那样具有表达和管理对象的能力，在所有数据操作之前，要先期定义对象，作为以后使用该对象的唯一标识。对象之间只有通过对象标识加以区分，并通过对象标识作为句柄继承或多重继承唯一标识过的对象。标识符通常是由系统自动生成的，不需要用户来完成这项工作。然而在使用这种功能时要注意：系统生成的标识符通常是特定于这个系统的，如果要将数据转移到另一个不同的数据库系统中，则标识符必须进行转化。同时对象标识符必须具有永久持久性，也就是说，特定对象一经产生，系统就赋予一个在全系统中唯一的对象标识符，应该是固定不变的，一直到它被删除。面向对象数据库系统必须具有生成对象标识并维护其永远不变性的机制。

8.1.2　面向对象数据模型

数据模型是现实世界对象或实体、以及对象的约束和对象间联系的逻辑组织。面向对象数据模型借鉴了面向对象的概念，是面向对象数据库系统所必须支持的数据模型。

面向对象数据库系统是以面向对象数据模型为基础的，是当今数据库技术发展的一大趋势。对于面向对象数据模型，已经有许多基本概念达成了共识，但是仍然缺少一个统一严格的定义。面向对象数据模型可以看作是一个更高层次上的实现数据模型的新成员，它经常被用作高层概念模型，尤其在软件工程领域中更是如此。一系列面向对象的概念构成了面向对象数据模型的基础。

1．知识点提炼

（1）对象结构

一般而言，一个对象可以对应着 E-R 模型中的一个实体。对象中封装的属性和方法对外界是不可见的，对象之间的相互作用要通过消息来实现。普遍来讲，一个对象有如下相关内容。

① 属性集合：一个对象的属性值构成了该对象的状态，类似于关系数据库中关系元组的属性。属性的值域可以是任何类，包括原子类，如整型值，字符串等。一个属性可以有一个单一值，也可以有一个来自于某个值域的值集，即一个对象的属性可以是一个对象，从而形成了嵌套关系。

② 方法集合：一个对象的方法作用于该对象的状态上，同一类对象所有操作的实现相同。方法包含定义和实现，定义规定了方法名称、参数的个数和类型、返回值的类型、以及可能的语义描述；实现是一段代码，用来实现方法的功能。方法的定义和实现是相互分离的，为程序员提供了极大的灵活性，甚至可以用不同的语言实现不同的操作。

③ 消息集合：消息是发送给对象以存取属性值的，除了通过对象所指定的公共界面外，没有其他方法可以访问该对象。对象接收外部传送的消息，执行相应的操作，操作的结果同样可以以消息的形式返回。

（2）对象类

在面向对象数据库中，类是一系列相似对象的集合，对应于 E-R 模型中的实体集概念。类是面向对象系统和数据库系统之间最重要的连接。首先，类直接说明了一个实例及其所属类之间的实例关系；其次，类提供了构成查询的基础；还有，类可以用来增加面向对象数据库的语义完整性；最后，类提出了所有对象的属性和方法的规格说明，便于生成对象。

每个对象是它所在类的一个实例。类的概念类似于关系模式，类的属性类似于关系模式中的属性；对象类似于元组，类的一个实例对象类似于关系中的一个元组。如果把类本身看作一个对象，则称之为类对象。与其相关的属性集和方法集适用于该类对象而不适用于该类的实例，这样的属性和方法称之为类属性和类方法。一个类的类属性常常用来描述该类的实例的聚集特性。例如，所有学生实例的“平均年龄”就是一个聚集特性的例子。

（3）继承与多重继承

在面向对象数据模型中，所有类形成了一个有限的层次结构或者是有根的无环有向图，称之为类层次。如有一个类 C 和一个连接到 C 的一组较低层类的集合 S，则集合 S 中的类称为类 C 的子类，而类 C 又称为集合 S 中类的父类。集合 S 中的任何类继承类 C 的所有属性和方法，并可以有自己定义的属性和方法。一个父类可以有多个子类，一个子类也可以有多个父类，都存在直接关联或者间接关联的现象。

在面向对象数据模型中存在着两种继承：继承（单继承）和多重继承。在大多数情况下，类的继承足以满足应用的要求，典型的树状结构组织用来表示类层次。在树状结构组织中，每个类最多有一个父类，即一个子类只能继承一个父类的属性、方法和消息。然而，有些情况用树状结构并不能很好地表达类层次。多重继承允许一个类从多个直接父类中继承属性、方法和消息，此时类层次可以用一个有向无环图来表示。

在图 8-3 中给出了一个学校数据库的类层次结构图，通过它分别来解释类层次、继承和多重继承。

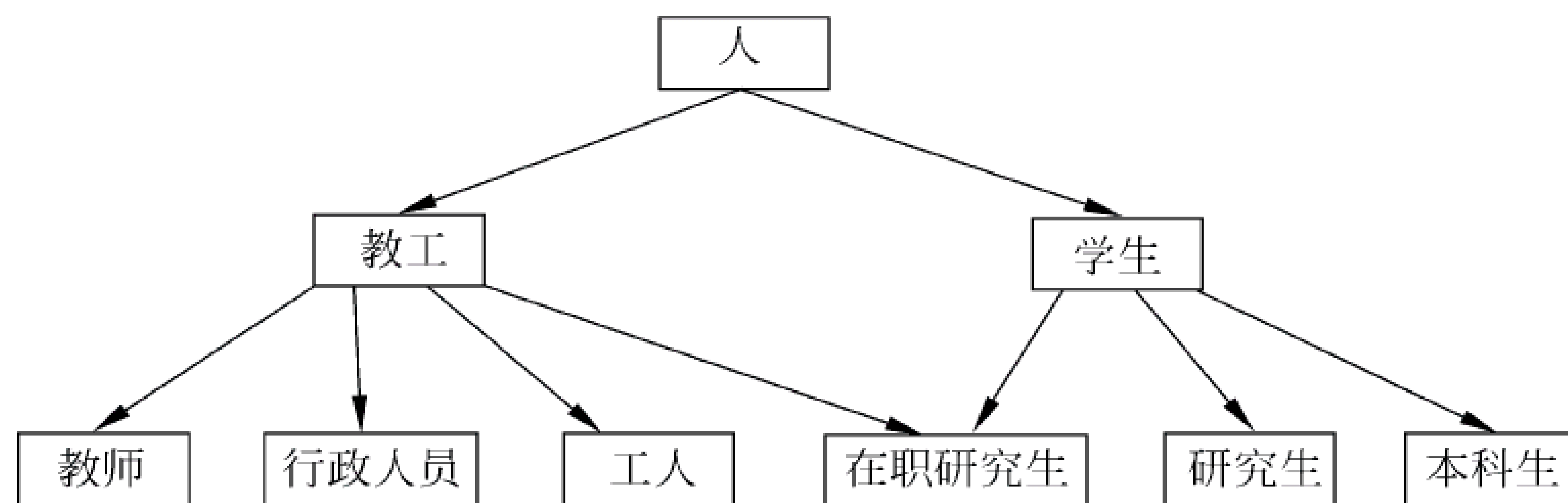


图 8-3　具有多继承的类层次结构图

在这个学校应用的面向对象数据库系统中，“人”是其他所有类的父类，它是这个有向无环图的根，是一个最高的类层次；在下面的一个类层次中，教工和学生是人的子类，它们继承了人的所有属性、方法和消息，同时又有本身的特殊属性、方法和消息；在最低的一个类层次中，教师、行政人员、工人和在职研究生是教工的子类，它们继承了教工和人的所有属性、方法和消息，在职研究生、研究生和本科生是学生的子类，它们继承了学生和人的所有属性、方法和消息。值得一提的是在职研究生既是教工的子类，也是学生的子类，它同时继承了教工和学生两个父类的所有属性、方法和消息。

类的继承带来很多的优点，子类在继承父类特性的同时，还可以定义自身的属性、方法和消息，但这样就可能和父类的属性、方法和消息发生冲突。这类冲突可能发生在子类和父类之间，通常由系统解决。对于子类和父类之间的同名冲突，一般是以子类定义的为准。但是在多继承中，一个子类可以有多个父类，如果这些父类中存在同名冲突，就会发生二义性。例如，教工和学生都有方法“显示信息”，它们共同的子类在职研究生就不知道应该继承哪一个方法了。在多继承中有 3 种处理二义性的方案：一是由用户选择继承的优

先次序；二是由系统指定继承某一个父类的定义；三是如果出现了二义性问题，就不允许多继承，甚至有些面向对象数据库系统根本不允许多继承。

（4）对象标识

每个对象有一个唯一的、由系统生成的对象标识（Object IDentifier，OID）。OID 的值对外部用户来说是不可见的，但是系统会在内部用这个值唯一的标识每个对象，并用这个值创建和管理内部对象引用。

相对于非面向对象数据模型和程序设计语言来说，对象标识给出了一种更强的标识概念，几种常用的标识形式如下所述。

① 值：用于标识的一个数据值。这种形式的标识常在关系数据库中使用，如一个元组的主码标识了这个元组。

② 名称：用于标识的用户提供的一个名称。在程序设计语言中，用户赋予每个变量一个名字来标识它；在文件系统中，用户给每个文件赋予一个名称来唯一的标识这个文件。

③ 内置名：以上两种标识是由用户给出的，而内置名则是一种由系统来提供的标识。这种形式的标识在数据模型或程序设计语言中使用。

（5）对象嵌套

对象嵌套是面向对象数据库系统中的一个重要概念。

在面向对象数据模型中，对象的一个属性可以是一个单一值，也可以是一个来自于值域的值集，即一个对象的属性可以是一个对象，形成了嵌套关系，产生了一个嵌套层次结构。

一个对象被称之为复杂对象，如果它的某个属性的值是另一个对象。复杂对象主要分为两类：非结构化的复杂对象和结构化的复杂对象。非结构化的复杂对象通常是数据库系统不明结构、需要大量存储空间的数据类型，如图像或大文本对象。结构化的复杂对象是指数据库系统清楚对象内部结构，并可以通过递归生成的对象。

2．难点分析

（1）对象标识符的持久性

不同的标识符其持久性程度是不同的，主要有以下几种。

① 过程内持久性：标识只有在单个过程的执行期间才是持久的，如过程内的局部变量。

② 程序内持久性：标识只有在单个程序或查询执行期间才是持久的，如程序设计语言中的全局变量、内存指针，SQL 语句中的元组标识符。

③ 程序间持久性：标识在从一个程序的执行到另一个程序的执行期间都保持不变，如指向磁盘上的文件系统数据的指针提供了程序之间的标识，SQL 语句中的关系名也具有程序间持久性。

④ 永久持久性：标识的持久性不仅仅跨越了各个程序的执行，还跨越了数据结构的重新组织。这种持久性正是面向对象系统所要求的。

但是面向对象数据库中的对象标识符必须具有永久持久性，也就是说，特定对象一经产生，系统就赋予一个在全系统中唯一的对象标识符，应该是固定不变的，一直到它被删除。面向对象数据库系统必须具有生成对象标识并维护其永远不变性的机制。

标识符通常是由系统自动生成的，不需要用户来完成这项工作。然而在使用这种功能时要注意：系统生成的标识符通常是特定于这个系统的，如果要将数据转移到另一个不同的数据库系统中，则标识符必须进行转化。而且，如果一个实体在建模时已经有一个系统之外的唯一标识符，则系统生成的标识符就可能是多余的，如身份证号码可以作为个人的唯一标识符。

早期的面向对象数据模型要求把所有的一切表示为对象，无论是一个简单的值还是一个复杂的对象，导致这样的情况出现：两个整型数值 10 和 20，需要创建两个具有不同 OID 的对象。这种模型需要生成很多的对象标识符，很不实用。因此，大多数的面向对象数据库系统允许有对象和值两种表示方法，即每个对象必须有一个永远不变的 OID，但是值没有 OID，值只是代表它自己。

（2）面向对象的数据库中为什么要引入对象嵌套

关系模式是对一个二维关系的描述，具有平面的结构。前面讲到的类层次结构形成了对象间的纵向关系，这里的对象嵌套层次结构则形成了对象间的横向关系。通过图 8-4 来说明。

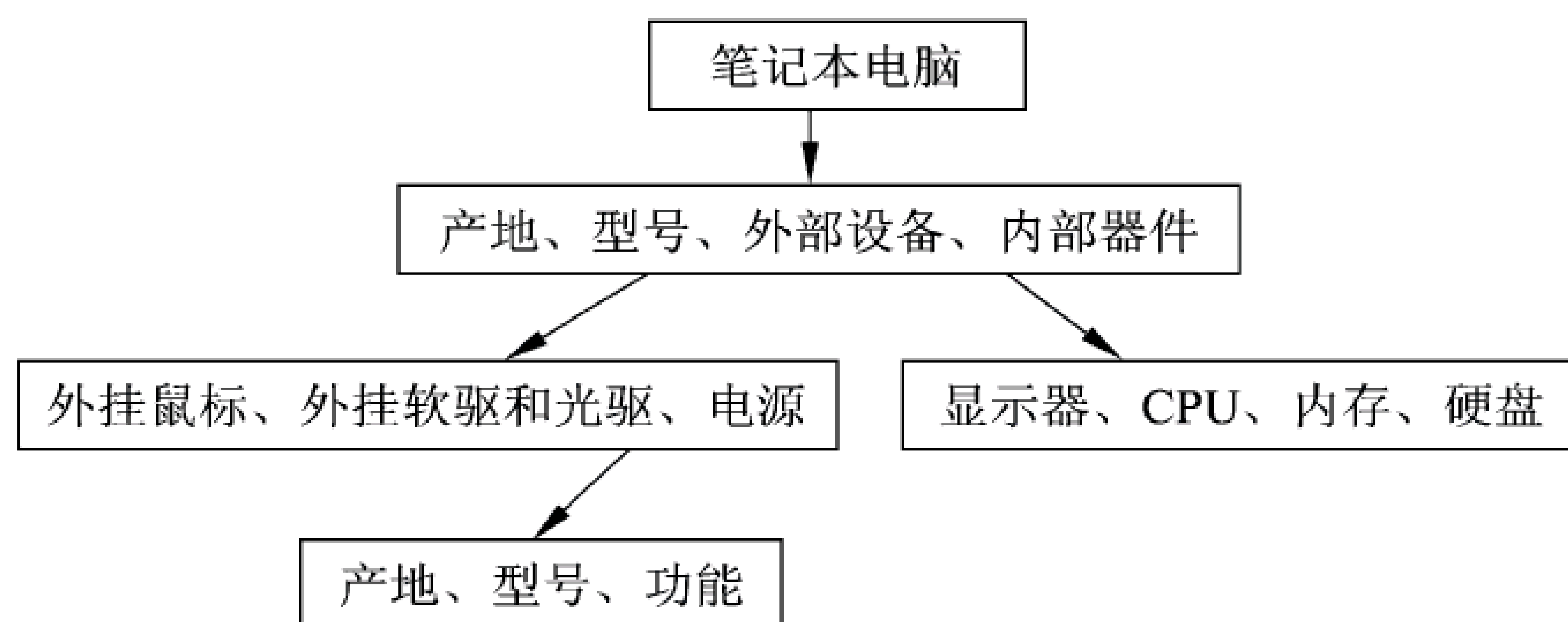


图 8-4　笔记本电脑的嵌套层次图

每台笔记本电脑包括产地、型号、外部设备和内部器件等属性。其中产地和型号的数据类型是字符串，外部设备和内部器件都不是标准数据类型，而是对象。外部设备包括外挂鼠标、外挂软驱和软驱、电源等属性；内部器件包括显示器、CPU、内存、硬盘等属性；电源也是一个对象，包括产地、型号、功率等属性。这样一种嵌套层次结构允许不用的用户采用不同的粒度来观察对象，突出了对象的特征，隐藏了不必要的信息，简化了查询。

一个对象如果它的某个属性的值是另一个对象，那么这个对象就属于复杂对象。复杂对象主要分为两类：非结构化的复杂对象和结构化的复杂对象。非结构化的复杂对象通常是数据库系统不明结构、需要大量存储空间的数据类型，如图像或大文本对象。结构化的复杂对象是指数据库系统清楚对象内部结构，并可以通过递归生成的对象。而引入对象嵌

套就是为了更清楚，直观地标识复杂对象内的逻辑关系。

3．典型例题

【例题 8-2】 面向对象数据库系统中的对象可以对应着 E-R 模型中的实体，一个对象结构有以下 3 种集合＿①＿、＿②＿、＿③＿。其中＿①＿构成了该对象的状态，＿②＿用于该对象的状态上，＿③＿作用是发送给对象以存取属性值。

① A．方法集合　B．属性集合　C．消息集合　D．值域集合

② A．方法集　B．属性集合　C．消息集合　D．值域集合

③ A．方法集合　B．属性集合　C．消息集合　D．值域集合

【答案】 ① B　② A　③ C

【解析】 一般而言，一个对象可以对应着 E-R 模型中的一个实体。对象中封装的属性和方法对外界是不可见的，对象之间的相互作用要通过消息来实现。普遍来讲，一个对象有如下相关内容：属性集合、方法集合、消息集合。

一个对象的属性值构成了该对象的状态，类似于关系数据库中关系元组的属性。属性的值域可以是任何类，包括原子类、如整型值、字符串等。一个属性可以有一个单一值，也可以有一个来自于某个值域的值集，即一个对象的属性可以是一个对象，从而形成了嵌套关系。

一个对象的方法作用于该对象的状态上，同一类对象所有操作的实现相同。方法包含定义和实现：定义规定了方法名称、参数的个数和类型、返回值的类型、以及可能的语义描述；实现是一段代码，用来实现方法的功能。方法的定义和实现是相互分离的，为程序员提供了极大的灵活性，甚至可以用不同的语言实现不同的操作。

消息是发送给对象以存取属性值的，除了通过对象所指定的公共界面外，没有其他方法可以访问该对象。对象接收外部传送的消息，执行相应的操作，操作的结果同样可以以消息的形式返回。

【例题 8-3】 面向对象数据模型可以看作是一个更高层次上的实现数据模型的新成员，它经常被用作高层概念模型，尤其在软件工程领域中更是如此，一系列面向对象的概念构成了面向对象数据模型的基础，其中包括对象结构、＿①＿、继承与多重继承、对象标识、和＿②＿。

① A．对象方法　B．对象属性　C．对象类　D．对象描述符

② A. 对象属性　B．持久化对象　C．对象嵌套　D．对象描述符

【答案】 ① C　② C

【解析】 数据模型是现实世界对象或实体、以及对象的约束和对象间联系的逻辑组织。面向对象数据模型借鉴了面向对象的概念，是面向对象数据库系统所必须支持的数据模型。其基本内容包括了对象结构、对象类、对象标识、继承与多重继承、对象嵌套。这些内容构成了面向对象数据库数据模型的基础。其中对象类是一系列相似对象的集合，对应于 E-R

模型中的实体集概念，而对象嵌套是由于复杂对象的引入而有某些面向对象语言衍生出的一种新特性，其用途在大本文数据库的存储，结构化信息的查询方面有着很强的支持作用。

8.1.3　面向对象数据库语言

目前面向对象数据库最大的障碍是缺乏统一的规范，各个数据库厂商有各自的访问接口。对象数据库比起关系数据库来，不只是基本的几种数据类型那么简单，它还涉及继承处理、多态等一大堆面向对象特征的实现，规范化道路当然困难重重。这也是对象数据库无法普及的一个重要原因。也有一些机构提出了一些建议的规范，比如制定 Corba 标准的 OMG 小组的一个分组对象数据库管理组织（Object Database Management Group，ODMG），该组织是面向对象数据库管理系统软件商的国际联盟，曾经提出一种标准，即 ODMG-93 或者 ODMG2.0。该标准已经修订为 ODMG3.0。ODMG 对象模型提供了数据类型、类型构造器、以及其他一些可以用于对象定义语言来说明对象数据库模式的概念，它是对象定义语言和对象查询语言的基础。

1．知识点提炼

（1）持久化程序设计语言（Persistent Programming Language）

数据库语言和传统的程序设计语言不同，在于直接操作持久数据。所谓持久数据是指创建这些数据的程序运行中止后依旧存在系统中，对现有的面向对象程序设计语言（OOPL）进行扩充，使之能处理持久数据，这样的 OOPL 称为持久话程序设计语言

（2）持久化程序设计语言的基本要素

① 对象的持久性。要把 OOPL 变成持久语言，第一步就是要提供一种办法，把对象区分成持久还是暂留的。在程序运行结束后，新创建的持久对象将被保留，而暂时对象将消失。

② 对象标识和指针。当一个持久对象被创建后，它就要被分配一个持久的对象标识符；当创建的对象为暂留时候，被分配一个暂留的对象标识符，在程序中止后，对象被删去，标识符失去意义。在 OOPL 的持久化程序语言中，持久对象的标识是以“持久化指针”实现的。与内存中的指针不一样，持久化指针在程序执行后及数据重组后仍然保持有效。程序员可以像使用内存中指针一样使用持久指针。在概念上，持久指针可以看作是数据库中指向对象的指针。

（3）持久对象的存储和访问

逻辑上，实现类的方法的程序代码应该和类的定义一起作为数据库模式的一部分存储，但现在往往将程序代码存储在数据库之外的文件中，目的是避免对编译器和 DBMS 软件进行集成。

查找数据库中对象的方法有以下 3 种。

第一种是根据对象名找到对象。实现时，每个对象有一个对象名，这个方法对少量的对象有效，但对大规模的对象存储就不实用了。

第二种是根据对象标识找对象，对象标识存储在数据库之外。

第三种是将对象按照聚集形式存放，然后利用程序循环找所需要的对象。聚集形式包括集合、多集等。

大多数商业 OODBS 都支持以上 3 种数据库访问形式。

（4）对象定义语言 ODL（Object Definition Language）

对象定义语言被设计成为支持 ODMG2.0 对象模型的语义结构，并且独立于任何特定的编程语言。它的主要用途是创建对象说明，也就是类和接口，因此对象定义语言不是一个完全的编程语言。用户可以独立于任何编程语言在对象定义语言中指定一种数据库模式，然后使用特定的语言绑定来指明如何将对象定义语言结构映射到特定编程语言的结构，如 C++、Smalltalk、Java。

（5）对象查数据询语言 OQL（Object Query Language）

ODMG2.0 还提出了对象查数据询语言（OQL）。对象数据查询语言被设计为与编程语言紧密配合使用，这些编程语言有一个 ODMG 绑定的定义，如 C++、Smalltalk、Java。这样嵌入某种编程语言的一个对象查询语言的查询，可以返回与那种语言的类型系统相匹配的对象。对于查询，对象查询语言语法和关系型标准查询语言 SQL 的语法相似，只是增加了有关对象的特征，如对象标识、复杂对象、操作、继承、多态性。

2．难点分析

（1）ODMG3.0 的改进

ODMG3.0 最大的改进之处是在于对 Java 语言的支持。ODMG3.0 将不再支持持久化 Java 程序语言，转而提出一个新的标准 JDO。对于 JDO 的详细情况超出本书大纲，将不再详叙。

（2）持久化语言与嵌入式语言的区别

① 在嵌入式语言中，宿主语言的类型系统与 SQL 的类型不同， 程序员要负责宿主语言与 DML 之间的类型转换。

而持久化程序设计语言的查询语言与宿主语言完全集成在一起，具有相同的类型系统，创建对象并将它存储在数据库中，不需要任何显示的类型转换或格式改变。任何格式转换对程序员都是透明的。

② 使用嵌入式查询语言的程序员要负责编写程序，把数据从数据库中取出放到内存中。在更新时候，程序员还需要编写程序段将更新过的数据回写到数据库。相比之下，在持久化程序设计语言中，程序员可以直接操作持久数据，而不必为存取数据编写程序。

3．典型例题

【例题 8-4】 作为面向对象数据库管理系统软件商的国际联盟——对象数据库管理组织 ODMG 提出了面向对象数据库语言的标准，它规定了面向对象数据库语言为__①__，其中__①__的基本要素包括了对象的持久性、__②__、和持久对象的存储和访问。

① A．持久化程序设计语言　　　　　　B．宿主语言

C．对象定义语言　　D．对象查数据询语言

（2）A．对象类型　　B．对象标识和指针

C．对象嵌套　　D．对象描述符

【答案】 ① A　② B

【解析】数据库语言和传统的程序设计语言不同，在于直接操作持久数据。所谓持久数据是指创建这些数据的程序运行中止后依旧存在系统中，对现有的面向对象程序设计语言（OOPL）进行扩充，使之能处理持久数据，这样的 OOPL 称为持久话程序设计语言

持久化程序设计语言的基本要素包括了对象的持久性、对象标识和指针持久、对象的存储和访问。其中当一个持久对象被创建后，它就要被分配一个持久的对象标识符，当创建的对象为暂留时候，被分配一个暂留的对象标识符，在程序中止后，对象被删去，标识符失去意义。在 OOPL 的持久化程序语言中，持久对象的标识是以“持久化指针”实现的。

与内存中的指针不一样，持久化指针在程序执行后及数据重组后仍然保持有效，程序员可以像使用内存中指针一样使用持久指针。在概念上，持久指针可以看作是数据库中指向对象的指针。

8.1.4　对象关系数据库系统

数据库系统面临着许多领域新的应用挑战，如音频和视频处理系统中的数字化信息，计算机辅助桌面排版系统中的大文本，人造卫星成像或天气预报中的图像，工程设计、生物基因组信息、建筑图中的复杂数据；股票市场交易历史或卖出历史中的时间序列数据；地图数据和业务数据中的空间和地理数据。显然需要设计某些数据库，它们可以开发、操纵和维护来自这些应用的复杂对象。

在面对上述复杂应用时，基本关系模型及其 SQL 语言的早期版本被证明是不适用的。层次数据模型可以很好地适用于在组织中自然存在的分层结构，但是它在数据中的内置层次路径上过于局限和固定。网状数据模型可以明确地对联系建模，但是在实现方面却需要使用大量的指针，而且不具备对象标识、继承、封装这类概念，也不支持多种数据类型和复杂对象。因此产生了一种趋势，即将对象数据模型中的特征和语言结合到关系数据模型中，这样扩展了关系数据模型，形成了对象关系数据库系统，使它能够处理当今复杂的应用。

对象关系数据模型扩展了关系数据模型的方式是通过提供一个包括复杂数据类型和面向对象的更丰富的类型系统。关系查询语言也需要做相应的扩展，以处理这些更丰富的类型系统。对象关系数据库系统是以对象关系数据模型为基础的，它为想要使用面向对象特征的关系数据库用户提供了一个方便的迁移途径。

1．知识点提炼

（1）嵌套关系

嵌套关系模型是关系模型的一个扩展，域可以是原子的也可以赋值为关系。这样元组在一个属性上的取值可以是一个关系，于是关系可以存储在关系中，从而形成了关系的嵌

套。这样一个复杂对象就可以用嵌套关系的单个元组来表示。如果将嵌套关系的一个元组视为一个数据项，在数据项和用户数据库观念上的对象之间就有了一一对应的关系。

以下为某剧组的演职人员设计一个嵌套的关系模式，其中包含了一个属性 Movies，它是一个代表所有该演职人员出演的电影集合的关系。Movies 的关系模式包括 Title、Year 和影片的长度 Length。关系 Star 的模式包括 Name、Address 和 Birthdate，还有就是 Movies 中的信息。另外，Address 属性也是一个关系模型（包含 Street 和 City 两个属性），可以在这个关系中记录演职人员的多个地址。Stars 的模式可以设计为：

```
Stars(Name, Address(Street, City), Birthdate,
      Movies(Title, Year, Length))
```

图 8-5 显示了嵌套关系 Stars 的一个例子，其中有两个元组，一个是 Fisher 的，另一个是 Hamill 的。

<table>
<tr><th rowspan="2">Name</th><th colspan="2">Address</th><th rowspan="2">Birthdate</th><th colspan="3">Movies</th></tr>
<tr><th>Street</th><th>City</th><th>Title</th><th>Year</th><th>Length</th></tr>
<tr><td rowspan="6">Fisher</td><td rowspan="3">Maple</td><td rowspan="3">H'wood</td><td rowspan="6">9/9/99</td><td rowspan="2">Star wars</td><td rowspan="2">1977</td><td rowspan="2">124</td></tr>
<tr></tr>
<tr><td rowspan="2">Empire</td><td rowspan="2">1980</td><td rowspan="2">127</td></tr>
<tr><td rowspan="3">Locust</td><td rowspan="3">Malibu</td></tr>
<tr><td rowspan="2">Return</td><td rowspan="2">1983</td><td rowspan="2">133</td></tr>
<tr></tr>
<tr><td rowspan="3">Hamill</td><td rowspan="3">Oak</td><td rowspan="3">B'wood</td><td rowspan="3">8/8/88</td><td>Star wars</td><td>1977</td><td>124</td></tr>
<tr><td>Empire</td><td>1980</td><td>127</td></tr>
<tr><td>Return</td><td>1983</td><td>133</td></tr>
</table>

图 8-5　演职人员以及他出演的电影的嵌套关系

在 Fisher 的元组中，可以看到名字是原子类型，接下来是 Address 的值。Address 是一个关系，这个关系有两个属性 Street 和 City，并有两个元组，每个对应于一个地址。然后是她的 Birthdate（它也是原子类型的）。最后是 Movies 的属性，它的类型是一个关系，这个关系有 Title、Year 和 Length 等 3 个属性，其值包含了 Fisher 最著名的 3 部影片。

第二个元组是 Hamill，其结构于上一个元组相同。其中 Address 关系只有一个元组，因为他只有一个住址。Hamill 的 Movies 关系的内容与 Fisher 的 Movies 关系内容很像，因为正巧两位演职人员的代表作一样。要注意的是这两个关系是元组的不同分量，只不过它们恰好有相同的值，这与两个不同分量恰好有相同的整数值 124 的情况类似。

（2）复杂类型

嵌套关系只是对基本关系模型扩展的一个实例，其他非原子数据类型，如嵌套记录，同样已被证明是有用的。面向对象数据模型已经导致了对于如对象的继承、引用等特征的需求。有了复杂对象系统和面向对象，能够直接表达 E-R 模型的一些概念，如实体标识、多值属性、一般化和特殊化，而不再需要经过关系模型的复杂转化。

通过对 SQL 的扩展，可以使用复杂类型。下面有关于复杂类型的一些简单概念加以介绍。

下面是对一个 books 表的定义：

```
CREATE TABLE books(
…
Keyword-Set SETOF (VARCHAR(20))
…
)
```

这个表中的 Keyword 属性比较特殊，因为它允许属性是集合。集合是集合体类型的一个实例，其他的集合体类型包括数组和多重集合。因此不同于普通关系数据库中表的定义，允许属性是集合，从而 E-R 图中的多值属性能够直接表述。

现在许多的数据库应用需要存储的属性很大，如一个人的照片，或者更大的，如高分辨率的医学图像或者录像剪辑。在 SQL:1999 中提供了新字符型数据大对象数据类型和二进制数据大对象数据类型。大对象一般用于外部的应用，通过 SQL 对它们进行全体检索是毫无意义的。取而代之，应用程序一般只检索大对象的定位器，然后用定位器从宿主语言中操作该对象。

下面说明结构类型的声明和使用：

```
CREATE TYPE MyString CHAR Varying
CREATE TYPE MyDate
(Day INTEGER,
Month CHAR(10),
Year INTEGER)
CREATE TYPE Document
(Name MyString,
Author-List SETOF(MyString),
Date MyDate,
Keyword-List SETOF(MyString))
CREATE TABLE Doc OF TYPE Document
```

第一个语句定义了一个类型 MyString，它是一个变长的字符串。第二个语句定义了一个类型 MyDate，它有 3 个组成部分：Date、month 和 Year。第三个语句定义了一个类型 Document，它包含一个 Name、一个作者的集合 Author-List、一个类型为 MyDate 的日期以及一个关键词集合。最后创建表 Doc，它包含了类型为 Document 的元组。上述表的定义与普通关系数据库中的表定义是有区别的，因为前者允许属性为集合或者如 MyDate 那样的属性具有结构类型，这些特征使得 E-R 图中的复合属性及多值属性能够直接表达。

（3）继承、引用类型

在这里的介绍是基于 SQL:1999 标准的，不过也会提到一些在这个标准中没有出现的，

但是在 SQL 标准的未来版本中会介绍到一些特征。

继承可以在类型的级别进行，也可以在表的级别上进行。首先考虑类型的继承。

假定有如下人的类型定义：

```
CREATE TYPE Person
(Name VARCHAR(20),
Address VARCHAR(20))
```

如果要在数据库中对那些是学生或教师的人分别存储一些额外的信息，由于学生和教师都是人，因而可以使用类型继承来定义学生和教师类型，如下：

```
CREATE TYPE Student
UNDER Person
(Degree VARCHAR(20),
Department VARCHAR(20))
CREATE TYPE Teacher
UNDER Person
(Salary INTEGER,
Department VARCHAR(20))
```

Student 和 Teacher 都继承了 Person 的属性，即 Name 和 Address。Student 和 Teacher 都被称为 Person 的子类型，Person 是 Student 的父类型，同时也是 Teacher 的父类型。

现在假定要存储关于助教的信息，这些助教既是学生又是教师，甚至可能是在不同的系里。如果类型系统支持多重继承，可以为助教定义一个类型，如下：

```
CREATE TYPE TeacherAssistant
UNDER Student, Teacher
```

TeacherAssistant 将继承 Student 和 Teacher 的所有属性，但是有一个问题，就是 Name、Address 和 Department 同时存在于 Student 和 Teacher 中。

Name 和 Address 属性实际上是从同一个来源即 Person 继承来的，因此同时从 Student 和 Teacher 中都继承这两个属性不会引起冲突。然而 Department 属性在 Student 和 Teacher 中分别都有定义，事实上，一个助教可能是某个系的学生同时又是另一个系的教师。为了避免两次出现 Department 之间的冲突，可以使用 AS 子句将它们重新命名，如下面的 TeacherAssistant 类型定义如下：

```
CREATE TYPE TeacherAssistant
UNDER Student WITH (Department AS Student-Dept),
Teacher WITH (Department AS Teacher-Dept)
```

在 SQL:1999 中只支持单继承，即一个类型只能继承一种类型，使用的语法如同前面

提到的例子。TeacherAssistant 例子中的多重继承在 SQL:1999 中是不支持的。

通过下面的例子来说明表继承。

假设定义 People 表如下：

```
CREATE TABLE People OF Person
```

那么再定义表 Students 和 Teachers 作为 People 的子表，如下：

```
CREATE TABLE Students OF Student
UNDER People
CREATE TABLE Teachers OF Teacher
UNDER People
```

子表的类型必须是父表类型的子类型，因此 People 中的每一个属性均出现在子表中。

当声明 Students 和 Teachers 作为 People 的子表时，每一个 Students 和 Teachers 中出现的元组也隐式存在于 People 中。所以，如果一个查询用到 People 表，它将查找的不仅仅是直接插入到这个表中的元组，而且还包含插入到它的子表 Students 和 Teachers 中的元组。然而，只有出现在 People 中的属性才可以被访问。

面向对象的程序设计语言提供了应用对象的能力，类型的一个属性可以是对一个指定类型的对象的引用。可以定义一个 Department 类型，它有一个 Name 字段和一个引用到 Person 类型的 Head 字段，然后定义一个 Department 类型的表 Departments，如下所示：

```
CREATE TYPE Department
(Name VARCHAR(20),
Head REF(Person) scope people)
CREATE TABLE Departments OF Department
```

在上面的定义中，使用关键字 SCOPE 来限定了引用范围。这里，引用限制在 People 表中的元组。

与复杂类型有关的查询，这里要介绍的是处理复杂类型的扩展 SQL 查询语言。与复杂类型有关的查询可以分为如下几类。

① 路径表达式。在 SQL:1999 中对引用取内容使用“→”符号。可以使用下面的查询来找出各个部门负责人的名字和地址：

```
SELECT Head→Name, Head→Address
FROM Departments
```

在上面的查询中，带有“→”符号的表达式被称为路径表达式。

② 以集合体为值的属性。如果想找出所有的码中包含 database 字样的书，如下查询即可：

```
SELECT Title
```

```
FROM Books
Where 'database' IN (UNNEST(Keyword-Set))
```

UNNEST（Keyword-Set）在无嵌套关系的 SQL 中相当于一个 SELECT-FROM-WHERE 的子表达式。

③ 嵌套与解除嵌套。将一个嵌套关系转换成为 1NF 的过程称为解除嵌套。关系 Doc 有 Author-List 和 Keyword-List 两个属性，这两者都是嵌套关系，同时关系 Doc 另外还有 Name 和 Date 两个属性，它们都不是嵌套关系。假定想要将该关系转化为单个平面关系，使其不包含嵌套关系或者结构类型作为属性，可以使用以下查询来完成这个任务：

```
SELECT Name, A AS Author, Date.Day, Date.Month, Date.Year, K AS Keyword
FROM Doc AS B, B.Author-List AS A, B.Keyword-List AS K
```

FROM 子句中的变量 B 被声明以 Doc 为取值范围，变量 A 被声明以该文档的 Author-List 中的作者为取值范围，同时 K 被声明以该文档的 Keyword-List 的关键词为取值范围。

反向过程即将一个 1NF 关系转化为嵌套关系称为嵌套。嵌套可以用对 SQL 分组的一个扩展来完成。在 SQL 分组的常规使用中，需要对每个组创建一个临时的多重集合关系，然后在这个临时关系上应用一个聚集函数。如果不应用聚集函数而只返回这个多重集合，就可以创建一个嵌套关系。假定有一个 1NF 关系 Flat-Doc，下面的查询在属性 Keyword 上对关系进行了嵌套：

```
SELECT Title, Author, (Date, Month, Year) AS Date, SET(Keyword) AS Keyword-List
FROM Flat-Doc
GROUP BY Title, Author, Date
```

（4）函数与过程

在对象关系数据库系统中允许用户定义函数与过程，它们既可以用某种数据操纵语言如 SQL 来定义，也可以通过外部的程序设计语言来定义，例如 Java、C 或 C++。有些数据库管理系统支持它们自己的过程语言，如 Oracle 中的 PL/SQL 和 Microsoft SQL Server 中的 Transact SQL，它们类似于 SQL 的有关过程的部分，但在语法和语义上有所区别，详细信息可参见各自的系统手册。

假设定义这样一个函数：给定一个文档，返回其作者的人数。可以定义这个函数如下：

```
CREATE FUNCTION Author-Count(One-Doc Document)
RETURN INTEGER AS
SELECT COUNT(Author-List)
FROM One-Doc
```

这里 Document 是一个类型名。这个函数用单个文档对象来调用，SELECT 语句同关系 One-Doc 一起执行，这个关系仅包括单个元组，即函数的参数。这个 SELECT 语句的结

果是单个值，严格来讲，它是一个只有单个属性的元组，其类型被转化为一个值。

上面的函数可以使用在如下查询中，该查询返回具有多于一个作者的所有文档的名称：

```
SELECT Name
FROM Doc
WHERE Author-Count(Doc)>1
```

注意，上面的 SQL 表达式中，尽管在 FROM 子句中 Doc 是指一个关系，但在 WHERE 子句中它隐含地被视为一个元组变量，因此它可以用来作为 Author-Count 函数的一个参数。

有些数据库系统允许我们使用如 C 或 C++这样的程序设计语言来定义函数。用这种方式定义的函数比用 SQL 定义的函数效率更高，并且能够执行有些无法用 SQL 完成的计算。使用这些函数的例子有很多，如在一个元组的数据上做一个复杂的算法。

用程序设计语言定义的函数在数据库系统的外部编译，它们需要被装入并与数据库系统代码一起执行。这个过程要冒一定的风险，因为程序中的错误可能会破坏数据库的内部结构，并且可能绕道数据库系统的存取控制功能。

使用程序设计语言定义的函数看起来与使用嵌入式 SQL 没什么不一样，使用嵌入式 SQL 时数据库查询包含在一个通用程序中，但是它们之间还是有一个重要差别。在嵌入式 SQL 中，用户程序将查询传送给数据库系统执行，结果以一次一个元组的形式返回给该程序。因此，用户书写的代码永远不会需要访问数据库本身，于是操作系统就可以保护数据库不被任何用户进程所存取。当在查询中使用用户编码的函数时，要么这些代码必须由数据库系统本身运行，要么该函数所操作的数据必须被复制到一个分离的数据空间中。第二种方法增加了系统开销，第一种方法则诱发了潜在的脆弱性，这同时表现在完整性方面和安全性方面。

2．难点分析

（1）嵌套关系与第一范式

在关系数据理论中定义了第一范式，它要求所有的属性都具有原子的域。原子域是指这个域中的元素是不可再分的单元。然而并非所有的应用都适用于第一范式关系建模。例如，某些应用的用户将数据库视为对象的一个集合，而不是记录的一个集合，这些对象可能需要数条记录来表示。一个简单、易用的界面要求用户直观概念上的一个对象与数据库系统概念上的一个数据项之间是一一对应的关系。由此，对象关系数据库引入了嵌套关系，解决了此类看似简单，但使用关系型数据库却比较难以解决的问题。

（2）面向对象与对象-关系

前面已经研究了建立在持久化程序设计语言上的面向对象数据库，也研究了建立在关系模型之上的面向对象的对象关系数据库。这两种类型的数据库系统在市场上都存在，数据库设计者要选择那种适合应用需求的系统。

程序设计语言的持久化扩展和对象关系系统有着不同的市场目标。SQL 语言的声明性特征和有限的能力为防止程序设计错误对数据造成破坏提供了很好的保护，同时使得一些高级优化，例如减少 I/O，变得相对简单。对象关系系统的目标在于通过使用复杂数据类型来简化数据建模和查询，典型的应用有复杂数据的存储和查询等。

然而，对于某些类型的应用，如主要在内存中运行和对数据库进行大批量访问的应用来说，一个说明性语言，SQL 会带来显著的性能损失。满足应用的高性能要求就是持久化程序设计语言的目标。持久化程序设计语言提供了对持久数据的低开销存取，并且取消了数据转换的要求。但是，持久化程序设计语言对由于程序错误而引起的数据破坏更为敏感，而且通常没有强大的查询能力。它们典型的应用包括 CAD 数据库。

这些不同种类的数据库系统的能力可以总结如下。

- 关系系统：简单数据类型、功能强大的查询语言、高保护性。
- 以持久化程序设计语言为基础的面向对象系统：复杂数据类型、与程序设计语言集成、高性能。
- 对象关系系统：复杂数据类型、功能强大的查询语言、高保护性。

这些描述具有普遍性，但是请记住对有些数据库系统来说它们的分界线是模糊的。例如，有些以持久化程序设计语言为基础的面向对象数据库系统是在一个关系数据库系统之上实现的，这些系统的性能可能比不上那些直接建立在存储系统之上的面向对象数据库系统，但这些系统却提供了关系系统所具有的较强的保护能力。

3．典型例题

【例题 8-5】 下列定义语句给出了有关教师（Teacher）、系（Department）和系主任（Director）信息的相关信息。

```
        CREATE  TYPE  MyString  CHAR  Varying;
          CREATE  TABLE  Department(University  MyString,
Dname  MyString,
STAFF  SETOFF(REF(Teacher)),
DIRE  REF(Director));
          CREATE  TABLE  Teacher(Tno  INTEGER,
Tname  MyString
Languages  SETOFF(MyString),
Countries  SETOFF(MyString),
Works_For  REF(Department));
          CREATE  TABLE  DIRECTOR(Dno  INTEGER)
                              Under  Teacher;
```

请使用 ORDB 的查询语言，分别写出下列查询的 SELECT 语句：

① 检索精通英语（English）的教师工号和姓名。

② 检索清华大学出访过日本（Japan）并且精通日语（Japanese）的系主任

【答案】

```
① SELECT  Tno, Tname
FROM  Teacher
WHERE  'English'  IN  Languages;
②  SELECT  D.Dno, D.Tname
        FROM  Director  AS  D
        WHERE  D.Works_FOR.University='Tinghua  University'
            AND  'Japan'  IN  D.Countries
            AND  'Japanese'  IN  D.languages;
```

【解析】 在有关教师（Faculty）、系（Department）和系主任（Director）信息的相关信息的定义语句中，第一个语句定义了一个类型 MyString，它是一个变长的字符串。第二个语句定义了一个类型 Department，它有 University、Dname。请注意在 Department 与表 Teacher 的嵌套关系，和 Director 的引用关系。最后创建表 Teacher，它包含了类型为 Department 的元组。上述表的定义与普通关系数据库中的表定义是有区别的，它包含了几种复杂关系，即 Teacher 表与表 Director 的关系，Teacher 表与 Department 类型的关系。所以我们在检索清华大学出访过日本（Japan）并且精通日语（Japanese）的系主任时，不仅仅查询了表 Teacher，同时由于表 Teacher 内的复杂关系，检索后得到的信息也包括表 Director 和类型 Department 相关信息。

8.2　ERP 和数据库

20 世纪 60 年代的制造业为了打破“发出订单，然后催办”的计划管理方式，设置了安全库存量，为需求与订货提前期提供缓冲。20 世纪 70 年代，企业的管理者们已经清楚地认识到，真正的需要是有效的订单交货日期，因而产生了对物料清单的管理与利用，形成了物料需求计划（MRP）。到了 20 世纪 80 年代，企业的管理者们又认识到制造业要有一个集成的计划，以解决阻碍生产的各种问题。要以生产与库存控制的集成方法来解决问题，而不是以库存来弥补或以缓冲时间的方法去补偿，于是 MRP-II 即制造资源计划产生了。20 世纪 90 年代以来，随着科学技术的进步及其不断向生产与库存控制方面的渗透，解决合理库存与生产控制问题需要处理大量的信息，同时企业资源的管理也更加复杂，这需要更高的信息处理效率。传统的人工管理方式难以适应以上系统，这时只能依靠计算机系统来实现。信息的集成度要求扩大到企业的整个资源的利用和管理，因此产生了新一代的管理理论与计算机系统，即企业资源计划（ERP）。图 8-6 根据大纲要求给出了企业资源计划和数据库的基本知识框架图。

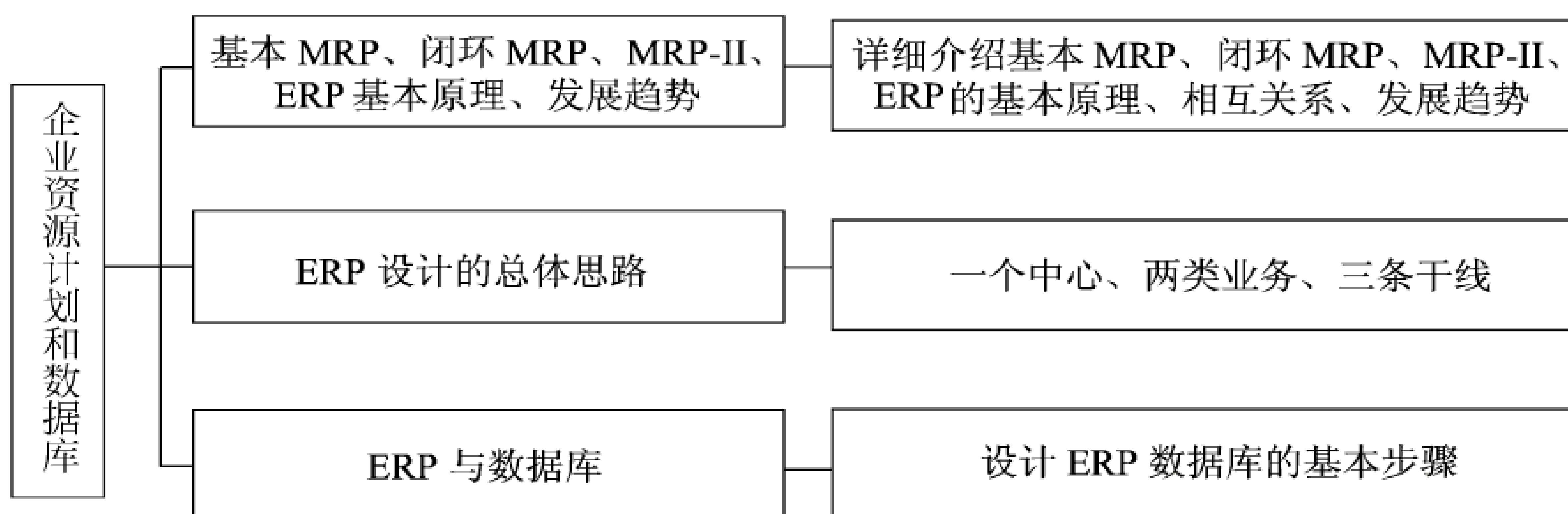


图 8-6　企业资源计划和数据库知识框架图

8.2.1　ERP 概述

ERP 是由美国 Garter Group Inc 咨询公司首先提出的，它是当今国际上先进的企业管理模式。其主要宗旨是对企业所拥有的人、财、物、信息、时间和空间等综合资源进行综合平衡和优化管理，面向全球市场，协调企业各管理部门，围绕市场导向开展业务活动，使得企业在激烈的市场竞争中全方位地发挥足够的能力，从而取得最好的经济效益。

1．知识点提炼

基本 MRP、闭环 MRP、MRP-Ⅱ、ERP 基本原理及发展历史。

（1）基本 MRP

20 世纪 40 年代初期，西方经济学家对库存物料随时间推移而被使用和消耗的规律进行了研究，提出了订货点的方法和理论，并将其运用于企业的库存计划管理中。20 世纪 60 年代中期，美国 IBM 公司的管理专家约瑟夫 • 奥利佛博士首先提出了独立需求和相关需求的概念，将企业内的物料分成独立需求物料和相关需求物料两种类型，并在此基础上总结出了一种新的管理理论：物料需求计划（Material Requirements Planning，MRP）理论，也称为基本 MRP。这种理论和方法与传统的库存理论和方法有着明显的不同，其最主要的特点是：在传统的基础上引入了时间分段和反映产品结构的物料清单（Bill Of Materials，BOM），从而较好地解决了库存管理和生产控制中的难题，即按时按量得到所需要的物料。基本 MRP 流程如图 8-7 所示。

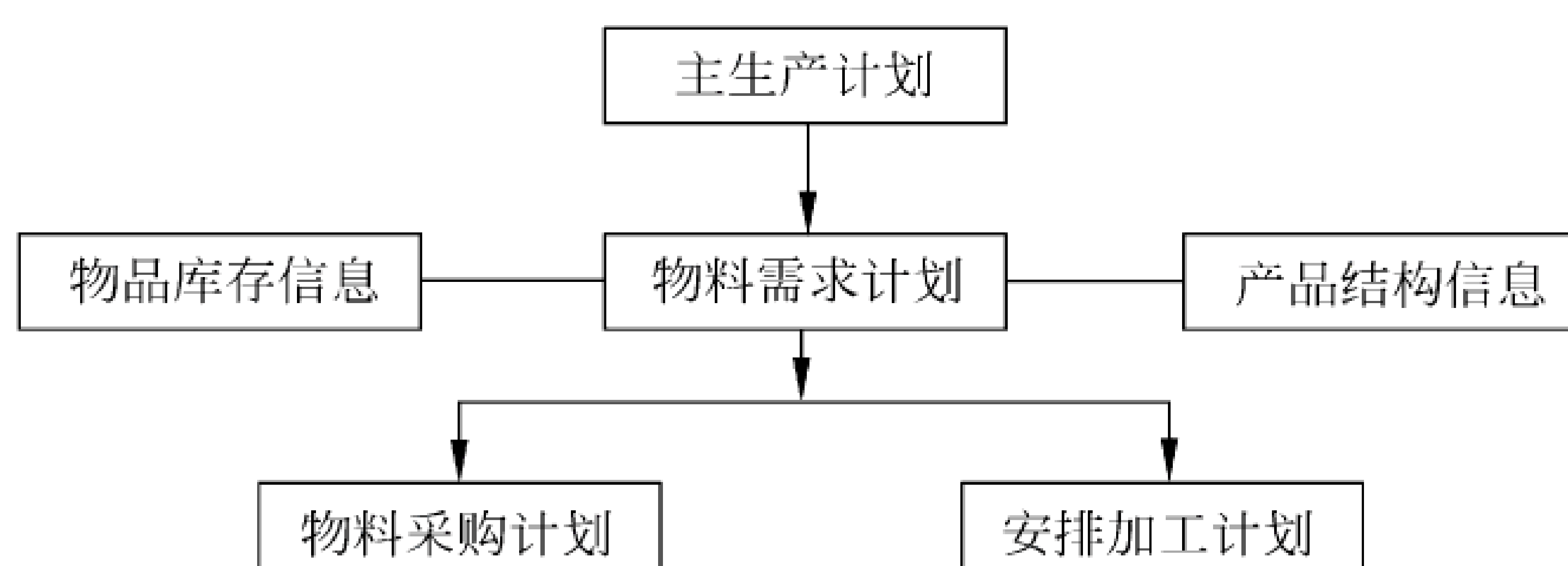


图 8-7　基本 MRP 的流程图

（2）闭环 MRP

在 MRP 的形成、制定过程中，考虑了产品结构相关信息和库存相关信息，但实际生产中的条件是变化的，如企业的制造工艺、生产设备及生产规模都是发展变化的，甚至要受社会环境的影响，如能源的供应、社会福利待遇等。基本 MRP 制定的采购计划可能受供货能力或运输能力的限制而无法保障物料的及时供应。另外，如果制定的生产计划未考虑生产线的能力，因而在执行时经常偏离计划，计划的严肃性将受到挑战。因此，利用基本 MRP 原理制定生产计划与采购计划往往不可行。因为信息是单向的，与管理思想不一致：管理信息必须是闭环的信息流，由输入至输出，再循环影响至输入端，从而形成信息回路。因此，随着市场的发展及基本 MRP 的应用与实践，20 世纪 80 年代初在此基础上发展形成了闭环 MRP 理论。闭环 MRP 的流程图如图 8-8 所示。

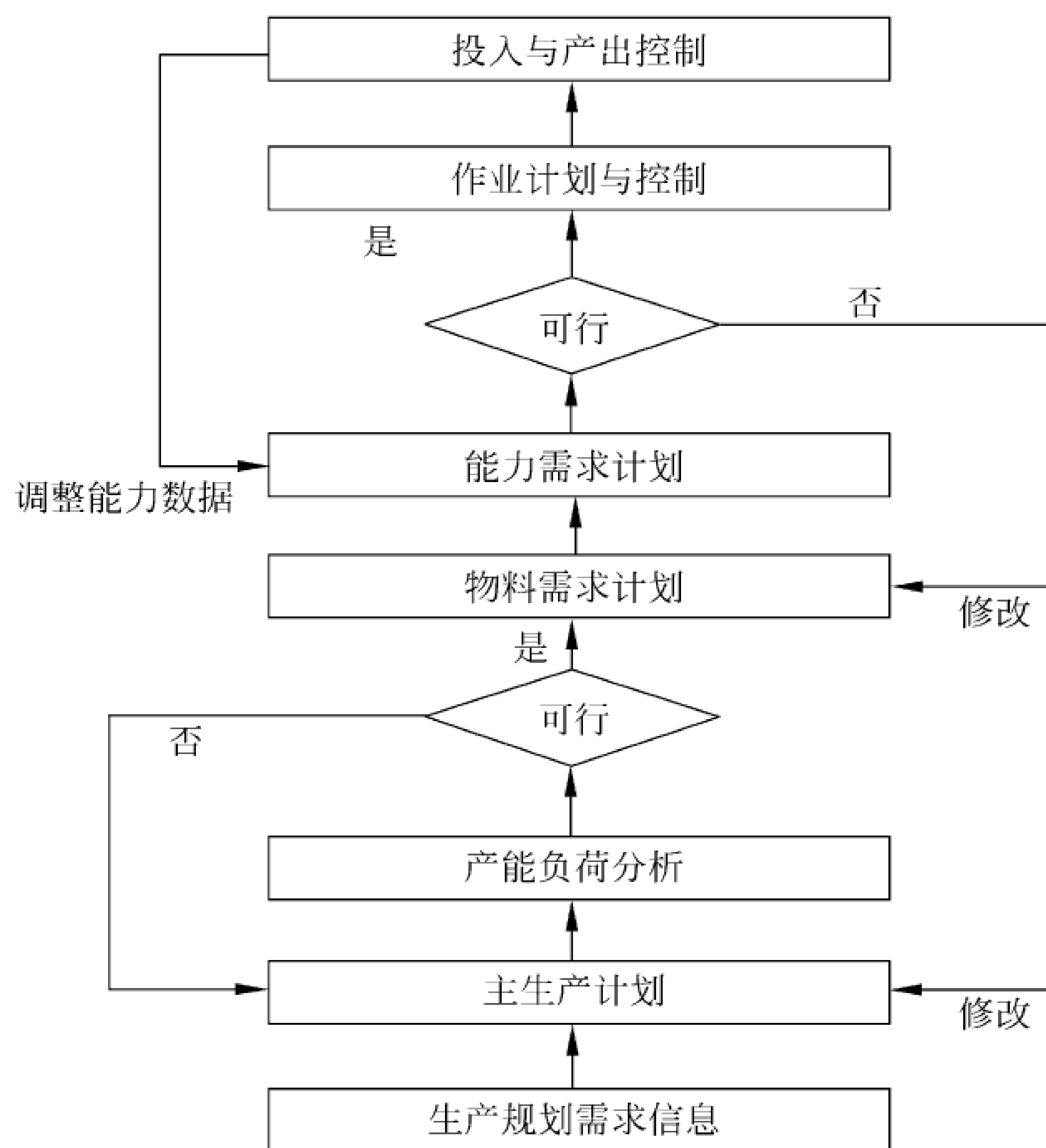


图 8-8　闭环 MRP 的流程图

闭环 MRP 理论认为主生产计划与物料需求计划应该是可行的，需要考虑能力的约束，换言之，即对能力提出需求计划。在能力允许的前提下，才能保证物料需求计划的执行和实现。在这种思想要求下，企业必须对投入与产出进行控制，也就是对企业的能力进行校检和执行控制。企业根据发展需要与市场需求来制定企业生产规划，根据生产规划制定主生产计划，同时进行生产能力与负荷的分析。该过程主要是针对关键资源的能力与负荷的

分析过程。只有通过对该过程的分析，才能达到主生产计划基本可靠的要求。再根据主生产计划、企业的物料库存信息、产品结构清单等信息来制定物料需求计划，由物料需求计划、产品生产工艺路线和车间各加工工序能力数据生成对能力的需求计划，通过对各加工工序的能力平衡，调整物料需求计划。如果这个阶段无法平衡能力，还有可能修改主生产计划。采购与车间作业按照平衡能力后的物料需求计划执行，并进行能力的控制，即输入输出控制，并根据作业执行结果反馈到计划层。因此，闭环 MRP 能较好地解决计划与控制问题，是计划理论的一次大飞跃，但它仍未彻底地解决计划与控制问题。

（3）MRP-II

MRP-II 围绕企业的基本经营目标，以生产计划为主线，对企业制造的各种资源进行统一计划和控制，使企业的物流、信息流和资金流畅通无阻，同时也实现了动态反馈。MRP-II 集成了应收、应付、成本及总账的财务管理。其采购作业根据采购单、供应商信息、收货单及入库单形成应付款信息（资金计划）；销售商品后，会根据客户信息、销售订单信息及产品出库单形成应收款信息（资金计划）；可根据采购作业成本、生产作业信息、产品结构信息、库存领料信息形成生产成本信息；能把应付款信息、应收款信息、生产成本信息和其他信息记入总账。产品的整个制造过程都伴随着资金的流通。通过对企业生产成本和资金运作过程的掌握，调整企业的生产经营规划和生产计划，因而得到更为可行、可靠的生产计划。图 8-9 给出了一个 MRP-II 系统通用的基本流程图。

前面讨论了基本 MRP、闭环 MRP 和 MRP-II 的理论，这些理论在相应的阶段都发挥了重要的作用，尤其是 MRP-II 的发展和应用。MRP-II 对世界的发展与应用产生了深远的影响。随着市场竞争日趋激烈和科技的进步，MRP-II 思想也逐步显示出其局限性，主要表现在以下几个方面。

① 企业竞争范围的扩大，要求在企业的各个方面加强管理，并要求企业有更高的信息化集成，要求对企业的整体资源进行集成管理，而不仅仅对制造资源进行集成管理。与竞争有关的物流、信息及资金要从制造部分扩展到全面质量管理、企业的所有资源（分销资源、人力资源和服务资源等）及市场信息和资源，并且要求能够处理工作流。在这些方面，MRP-II 都已经无法满足。

② 企业规模不断扩大。多集团、多工厂要求协同作战已超出了 MRP-II 的管理范围。全球范围内的企业兼并和联合潮流方兴未艾，大型企业集团和跨国集团不断涌现，企业规模越来越大，这就要求集团与集团之间，集团内多工厂之间统一计划，协调生产步骤，汇总信息，调配集团内部资源。这些既要独立，又要统一的资源共享管理是 MRP-II 目前无法解决的。

③ 信息全球化趋势的发展要求企业之间加强信息交流和信息共享。企业之间既是竞争对手，又是合作伙伴。信息管理要求扩大到整个供应链的管理，这些更是 MRP-II 所不能解决的。

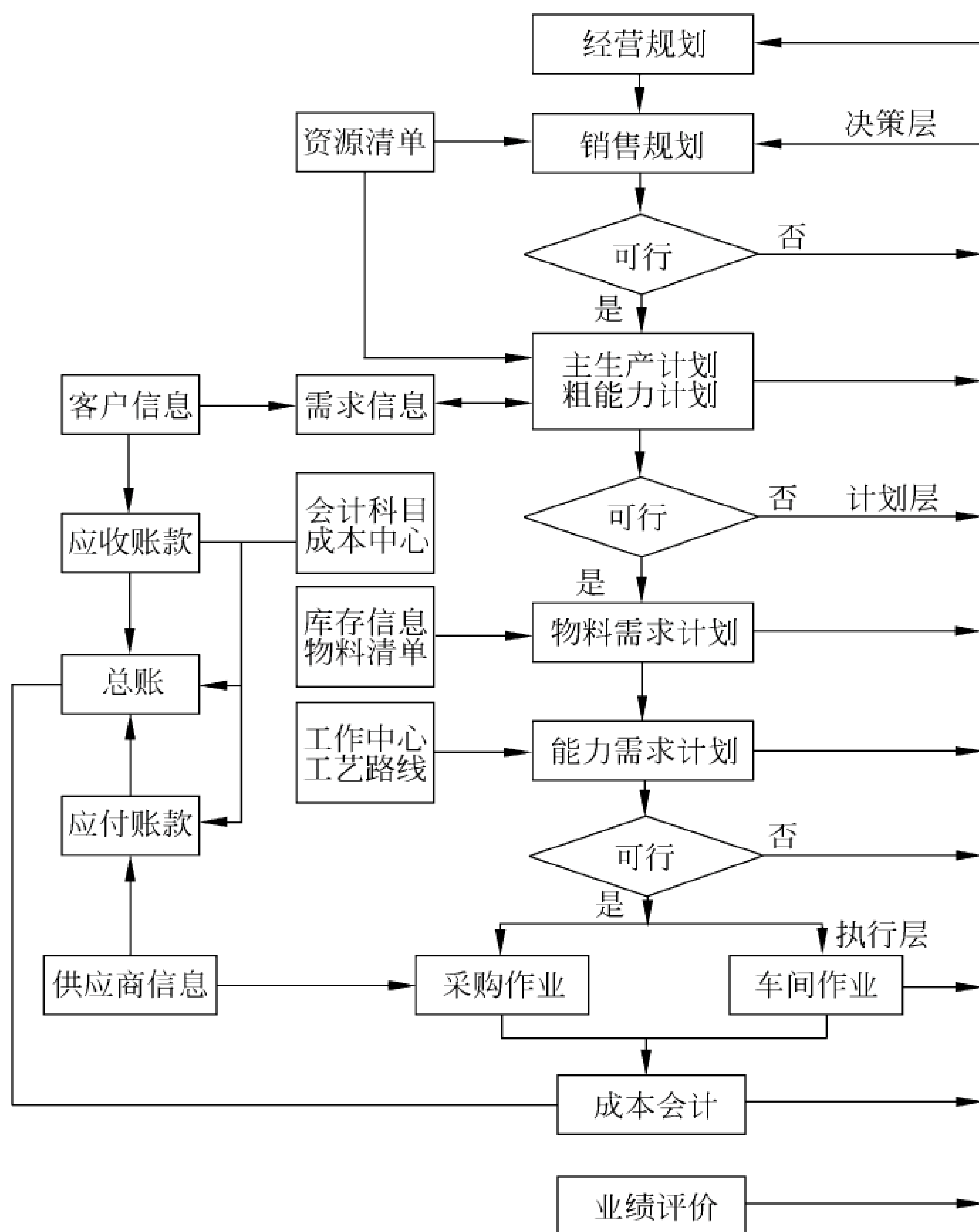


图 8-9　MRP-II 的流程图

（4）企业资源计划的基本原理与发展趋势

企业的所有资源可以概括为 3 大流：物流、资金流和信息流。企业资源计划（Enterprise Resource Planning，ERP）也就是对这 3 种资源进行全面集成管理的管理信息系统。概括地说，ERP 是建立在信息技术基础上，利用现代企业的先进管理思想，全面地集成了企业的所有资源信息，并为企业提供决策、计划、控制与经营业绩评估的全方位和系统化的管理。图 8-10 给出了一个 ERP 系统通用的基本流程图。

（5）ERP 设计的总体思路

ERP 设计的总体思路即把握一个中心、两类业务、三条干线。

① 一个中心

企业的主要目的是赢利，因而企业的每个业务活动都要考虑企业的经营目标，都会有输入的费用和输出的业务结果。因此，各项业务活动和功能模块要考虑归集到财务的数据，财务应是各项业务的归集中心，这是在系统规划与设计实现时必须考虑到的。同时财务的

处理要考虑本国国情，使从其他模块传递到财务的数据符合财务制度要求，可以为财务所利用，保证各个模块与财务之间数据传递的有效和畅通，实现财务业务处理的高度集成。

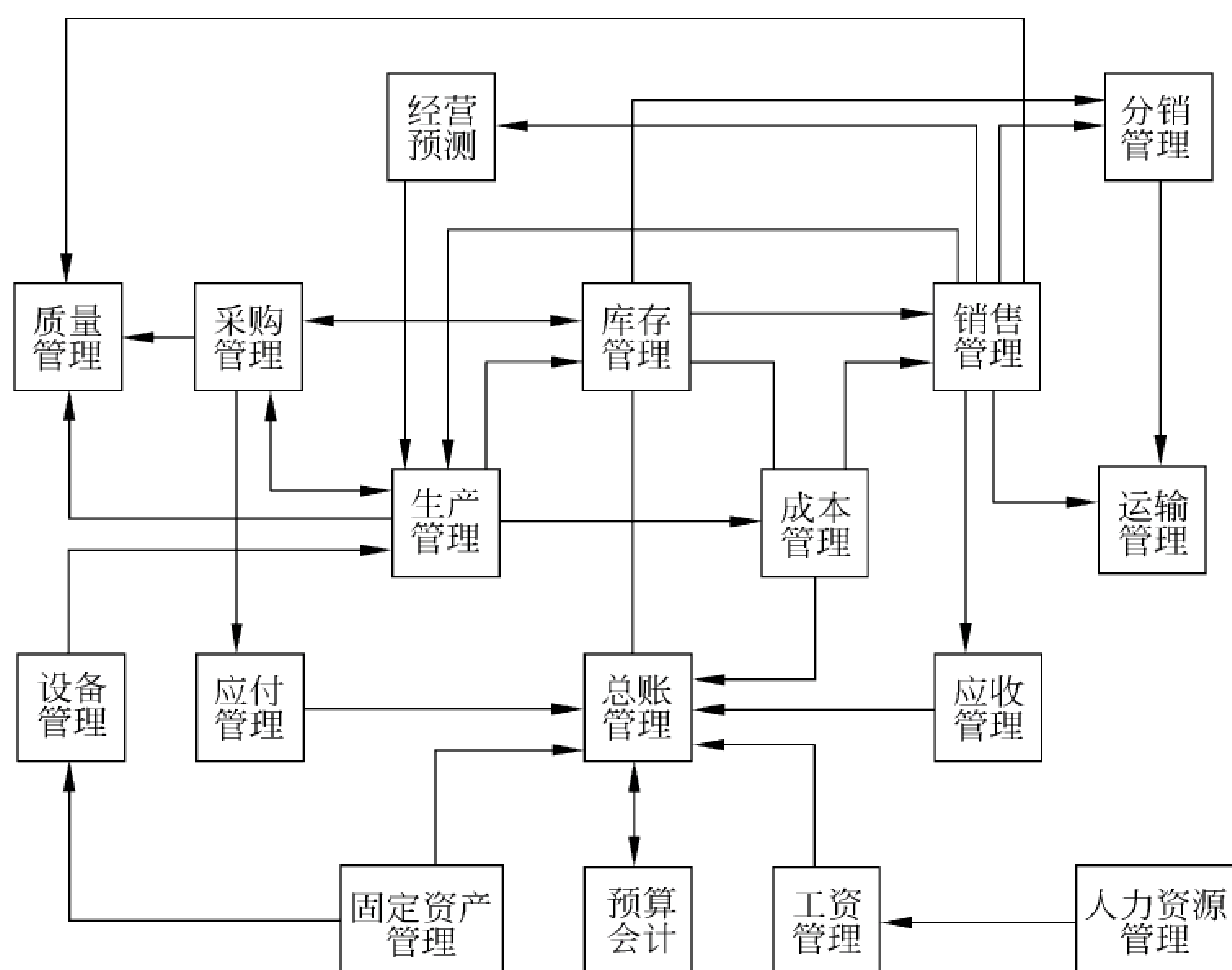


图 8-10　ERP 的基本流程图

② 两类业务

从 ERP 原理可以得出这样一个结论：计划与执行贯穿了系统的整个过程。从计划到执行计划，再反馈到计划层，影响计划的制定和修正，这个过程周而复始，形成一个闭环，也体现了管理的闭环原则。各个模块的业务处理，围绕计划展开，计划有经营规划、销售计划、主生产计划、采购计划、资金需求计划、车间作业计划（生产、检验）等。

③ 三条干线

ERP 设计的三条干线为供应链管理、生产管理、财务管理。这 3 条干线也就是制造业业务处理的主流业务，因而在进行设计规划与设计实现时要围绕这 3 条干线进行分工和协调，其过程分述如下。

- 供应链管理是企业物流业务的主干线，它处理企业从原材料供应，产品存储到产品销售的整个流程，其物流管理的核心是库存的管理，并要综合考虑整个物流供应链的管理。该过程的主要信息和数据有物品代码资料、物品库存资料、供需方的资料等，其中销售计划、合同和订单是主生产计划的入口数据。
- 生产业务是制造业的主体业务，包含主生产计划的制定、资源的利用、下达生产计划和生产作业的控制等业务。这个过程的运作涉及很多企业的重要基础数据，

如产品结构清单、工艺路线、工作中心资源与能力等，在设计时要尽可能地考虑各个行业的不同产品结构特点、工艺特点和业务管理特点等。

- 财务管理。财务集成设计是最终完成 ERP 集成的关键，是企业各项业务活动最终结果的体现，也是一个中心的最终体现。

这 3 条干线的数据相互利用，业务互相联系、渗透。因此，做好数据库的设计非常重要，它直接关系到集成的好坏和系统效率的高低。ERP 数据库设计时应尽量将一个实体的描述放入一个数据库中，如在设计基础数据采集入口时，规定一种数据只从一个入口录入，其他地方只是使用数据或继续补充，不再重复录入，这样就可以避免系统中对一个事件的描述存在数据不一致的问题，达到了数据只录入一次然后充分共享的目的，既减少了录入数据的工作量，又降低了出错的几率。利用数据库的约束规则可以做到这一点，这涉及了数据库的数据处理效率问题。

围绕这 3 条干线的模块划分如下所述。

- 进销存管理模块系列，包括库存管理、销售管理、采购管理及分销资源计划管理等。
- 生产管理模块系列，包括制造标准、主生产计划、物料需求计划、能力需求计划、车间作业管理、重复制造生产管理、质量管理及设备管理等。
- 财务管理模块系列，包括总账管理、应收账款管理、应付账款管理、预算会计、现金管理、账簿报表管理、固定资产管理、工资管理及成本会计等。
- 其他补充模块，如人力资源管理、技术管理、经营预测系统、决策系统和系统管理、工作流管理等等。

2．难点分析

（1）ERP 与 MRP-II 关系

ERP 理论与系统是从 MRP-II 发展而来的，它除继承了 MRP-II 的基本思想（制造、供销及财务）外，还大大地扩展了管理的模块，如多工厂管理、质量管理、设备管理、运输管理、分销资源管理、过程控制接口、数据采集接口、电子通信等模块。它融合了离散型生产和流程型生产的特点，扩大了管理的范围，更加灵活地开展业务活动，实时地响应市场需求。它还融合了多种现代管理思想，进一步提高了企业的管理水平和竞争力。因此 ERP 理论不是对 MRP-II 理论的否认，而是继承与发展。MRP-II 的核心是物流，主线是计划，伴随着物流的过程，同时存在资金流和信息流。ERP 的主线也是计划，但 ERP 已将管理的重心转移到财务上，在企业整个经营运作过程中贯穿了财务成本控制的概念。总之，ERP 极大地扩展了业务管理的范围及深度，包括质量、设备、分销、运输、多工厂管理、数据采集接口等。

为了便于对 MRP/MRP-II/ERP 各系统有一个完整、清晰的认识，下面将它们之间的关系用一个简单的包含图来表示，如图 8-11 所示。

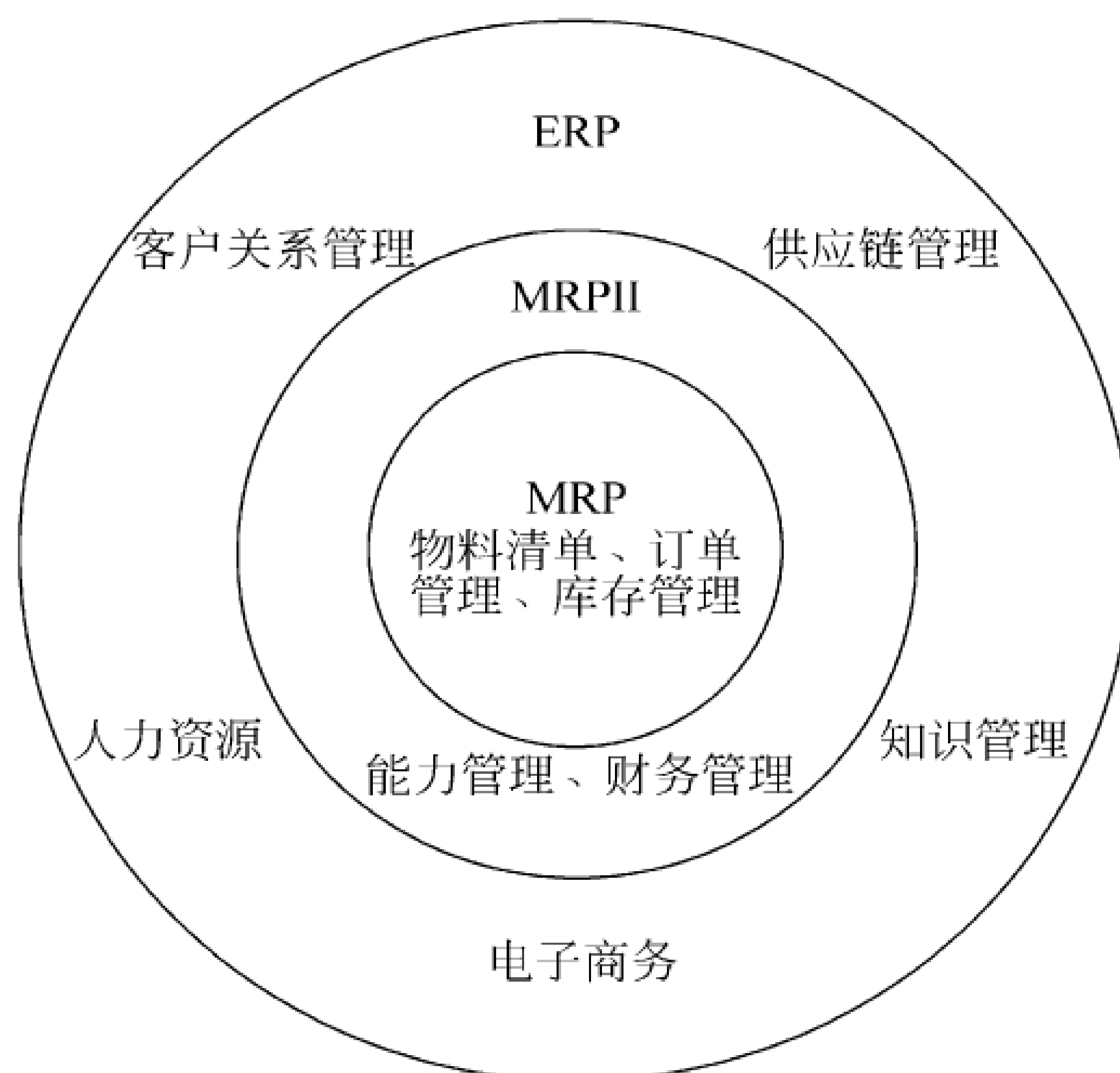


图 8-11 MRP/MRP-II/ERP 关系图

（2）ERP 未来的发展趋势进行展望

① 管理范围更加扩大

ERP 的管理范围有继续扩大的趋势，继续扩充供需链管理（Supply Chain Management，SCM），SCM 融合了企业本身的所有经营业务、企业的办公业务、企业之间的协同商务业务等，如电子商务、客户关系管理、办公室自动化等。协同商务是指企业内部人员，贯穿于贸易共同体的业务伙伴和客户之间的协作及电子化的业务交互过程。贸易共同体可以是一个行业或行业分支，也可以是供应链或供应链的一部分。此外，ERP 系统还日益和计算机辅助设计、计算机辅助工艺设计、产品数据管理、POS 系统以及自动货仓等系统融合。这样企业管理人员在办公室中完成的全部业务都被纳入到了管理范围内，实现了对企业的所有工作及相关内外环境的全面管理。

② 继续支持与扩展企业的流程重组

企业的外部与内部环境变化是相当快的。企业要适应这种快节奏的变化，就要不断地调整组织机构和业务流程。因此，ERP 的发展必然要继续支持企业的这种变化，使企业的工作流程能够按照业务的要求进行组织，以便集中相关业务人员，用最少的环节，最快的速度和最经济的形式，完成某项业务的处理过程。

③ 运用最先进的计算机技术

信息是企业管理和决策的依据，计算机系统能够及时而准确地为企业提供必要的信息，因此 ERP 的发展是离不开先进的计算机技术的。Internet 和 Intranet 技术，使企业内部及企业与企业之间的信息传递更加畅通。面向对象技术的发展使企业内部的重组变得更加快捷和容易。计算机在整个业务过程中产生信息的详尽记录与统计分析，使决策变得更加科学和有目的性。新的计算机技术的不断涌现为 ERP 的发展提供了广阔的前景。

3．典型例题

【例题 8-6】企业信息化的发展历经了 MRP 闭环的 MRP ，MRP-Ⅱ和 ERP，其中 MRP 着重管理企业的__①__，接着又发展成闭环的 MRP，闭环 MRP 强调了__②__。当发展到 MRP-Ⅱ阶段的时候，围绕着__③__。以__④__为主线直到上世纪 90 年代提出 ERP 理论，才真正的将__⑤__这三者紧密结合在一起。

① A．供应链计划　　B．物料需求计划
　C．物流管理计划　　D．销售管理计划

② A．计划的可实施性　　B．生产能力对需求计划的影响
　C．销售计划　　D．生产管理

③ A．企业的基本经营目标　　B．物料需求计划
　C．物流管理计划　　D．销售管理计划

④ A．计划的可实施性　　B．销售计划
　C．生产计划　　D．物料计划

⑤ A．供应链管理、生产管理、财务管理
　B．客户关系管理、生产管理、财务管理
　C．供应链管理、生产管理、人力资源管理
　D．供应链管理、销售管理、财务管理

【答案】 ① B　② B　③ A　④ C　⑤ A

【解析】 物料需求计划（Material Requirements Planning，MRP）理论，也称为基本 MRP。这种理论和方法与传统的库存理论和方法有着明显的不同，其最主要的特点是在传统的基础上引入了时间分段和反映产品结构的物料清单（Bill Of Materials，BOM），从而较好地解决了库存管理和生产控制中的难题，即按时按量得到所需要的物料。

闭环 MRP 理论认为主生产计划与物料需求计划应该是可行的，需要考虑能力的约束，换言之，即对能力提出需求计划。在能力允许的前提下，才能保证物料需求计划的执行和实现。

MRP-II 围绕企业的基本经营目标，以生产计划为主线，对企业制造的各种资源进行统一计划和控制，使企业的物流、信息流和资金流畅通无阻，同时也实现了动态反馈。

ERP 设计的三条干线为供应链管理、生产管理、财务管理。这 3 条干线也就是制造业业务处理的主流业务，因而在进行设计规划与设计实现时要围绕这 3 条干线进行分工和协调，其过程分述如下。

① 供应链管理是企业物流业务的主干线，它处理企业从原材料供应，产品存储到产品销售的整个流程，其物流管理的核心是库存的管理，并要综合考虑整个物流供应链的管理。该过程的主要信息和数据有物品代码资料、物品库存资料、供需方的资料等，其中销售计划、合同和订单是主生产计划的入口数据。

② 生产业务是制造业的主体业务，包含主生产计划的制定、资源的利用、下达生产计划和生产作业的控制等业务。这个过程的运作涉及很多企业的重要基础数据，如产品结构清单、工艺路线、工作中心资源与能力等，在设计时要尽可能地考虑各个行业的不同产品结构特点、工艺特点和业务管理特点等。

③ 财务管理。财务集成设计是最终完成 ERP 集成的关键，是企业各项业务活动最终结果的体现，也是一个中心的最终体现。

【例题 8-7】 ERP 设计的总体思路即把握一个中心、两类业务、三条干线，其中一个中心为__①__，它围绕着两类业务__②__，三条干线分别为供应链管理、生产管理、财务管理。

① A．以生产为中心　B．以销售为中心　C．以盈利为中心　D．以客户为中心

② A．计划和执行　B．生产和销售　C．客户与企业　D．管理与经营

【答案】 ① C　② A

【解析】 ERP 设计的总体思路即把握一个中心、两类业务、三条干线。

① 一个中心

企业的主要目的是赢利，因而企业的每个业务活动都要考虑企业的经营目标，都会有输入的费用和输出的业务结果。因此，各项业务活动和功能模块要考虑归集到财务的数据，财务应是各项业务的归集中心，这是在系统规划与设计实现时必须考虑到的。同时财务的处理要考虑本国国情，使从其他模块传递到财务的数据符合财务制度要求，可以为财务所利用，保证各个模块与财务之间数据传递的有效和畅通，实现财务业务处理的高度集成。

② 两类业务

从 ERP 原理可以得出这样一个结论：计划与执行贯穿了系统的整个过程。从计划到执行计划，再反馈到计划层，影响计划的制定和修正，这个过程周而复始，形成一个闭环，也体现了管理的闭环原则。各个模块的业务处理，围绕计划展开，计划有经营规划、销售计划、主生产计划、采购计划、资金需求计划、车间作业计划（生产、检验）等。

③ 三条干线

ERP 设计的三条干线为供应链管理、生产管理、财务管理。这三条干线也就是制造业业务处理的主流业务，因而在进行设计规划与设计实现时要围绕这三条干线进行分工和协调。

8.2.2　ERP 与数据库

在 ERP 系统中，数据流贯穿了整个系统的不同模块。ERP 就是结合企业已有的数据，更好地管理它们在金融、需求、物流、人力资源和订货方面的经营活动。其目的是降低成本、压缩运送时间、减少冗员和存货、改进客户服务。为了达到提高利润的目的，公司首先必须从其客户和销售活动中积累信息，并把它们存入一个不断更新的数据库。

ERP 系统可以是传统的 C/S 结构，也可以是现在流行的 B/S 模式的三层结构。前台使用的开发环境和后台的数据库也不尽相同。在实际的 ERP 系统开发中，经常要同时对多个数据库操作，一般是一个主表带多个从表，主表的某一个字段和从表的某一字段关联，而主表该字段的数据要从对应从表中获取，多表操作可以归纳为主从表操作。而每个主表或子表又同时面对若干数据流的访问和操作，所以 ERP 运行数据库的成功、高效与否，直接取决于 ERP 数据模型的建立和数据库的设计。

为了讨论运行数据库与 ERP 数据模型之间的关系，以 ERP 系统中最为典型的销售管理为例给出如何根据企业需求设计出合理的数据模型来满足运行数据库。

（1）销售管理业务分析

企业的销售管理工作主要由企业的销售部门完成，销售部门与生产部门，财务部门和仓库管理部门有着密切的业务联系。

- ❑ 销售部门制定销售预测、计划或客户订单后，将产品订货和交货情况汇总通知计划部门或生产部门作成生产计划；
- ❑ 生产部门根据计划安排领料生产，进入生产作业控制，产品完工后进入库管处理（按照订单或加工单入库）；
- ❑ 仓库部门按照计划发料、安排产品入库，并按照出货通知（根据订单的交货期）组织出货，产生出、入库单据教财务部门；
- ❑ 财务部门根据仓库的出入库单据、出货发票作账；客户收到货物和结算发票后付款给企业的财务部门；
- ❑ 销售部门记录有关的售前、售中、售后服务情况，并对有关的质量问题提交给质量管理部门进行产品质量分析。

把销售业务子系统管理分为基本资料管理和业务处理两大部分，这样设计的考虑是处于运行数据库的优化设计。

（2）销售管理业务数据流图

图 8-12 是企业销售管理业务的第一层数据流程图，图中将销售管理分为销售基础数据管理，销售计划管理，销售订单管理，销售收发货管理和销售服务管理 5 个子系统。

在对 5 个子系统作进一步的分析后，得出了销售管理业务的第二层数据流（图 8-13～图 8-17），第二层数据流注重的实际工作与信息的数据流的直接映射。请注意，在很多流程的开始，基础数据的获取可以分为数据录入和外来数据文件的数据载入。

（3）销售管理业务 E-R 关系图

销售管理业务 E-R 关系图，如图 8-18 所示。

（4）销售管理功能模块图

图 8-19 总结了前面的企业销售管理 E-R 图及各二级数据流程图，给出了销售管理功能模块图，这些图就是建立企业销售管理的数据库基表的根据。限于篇幅的原因，不能一一对应地将功能模块与数据库表做映射。

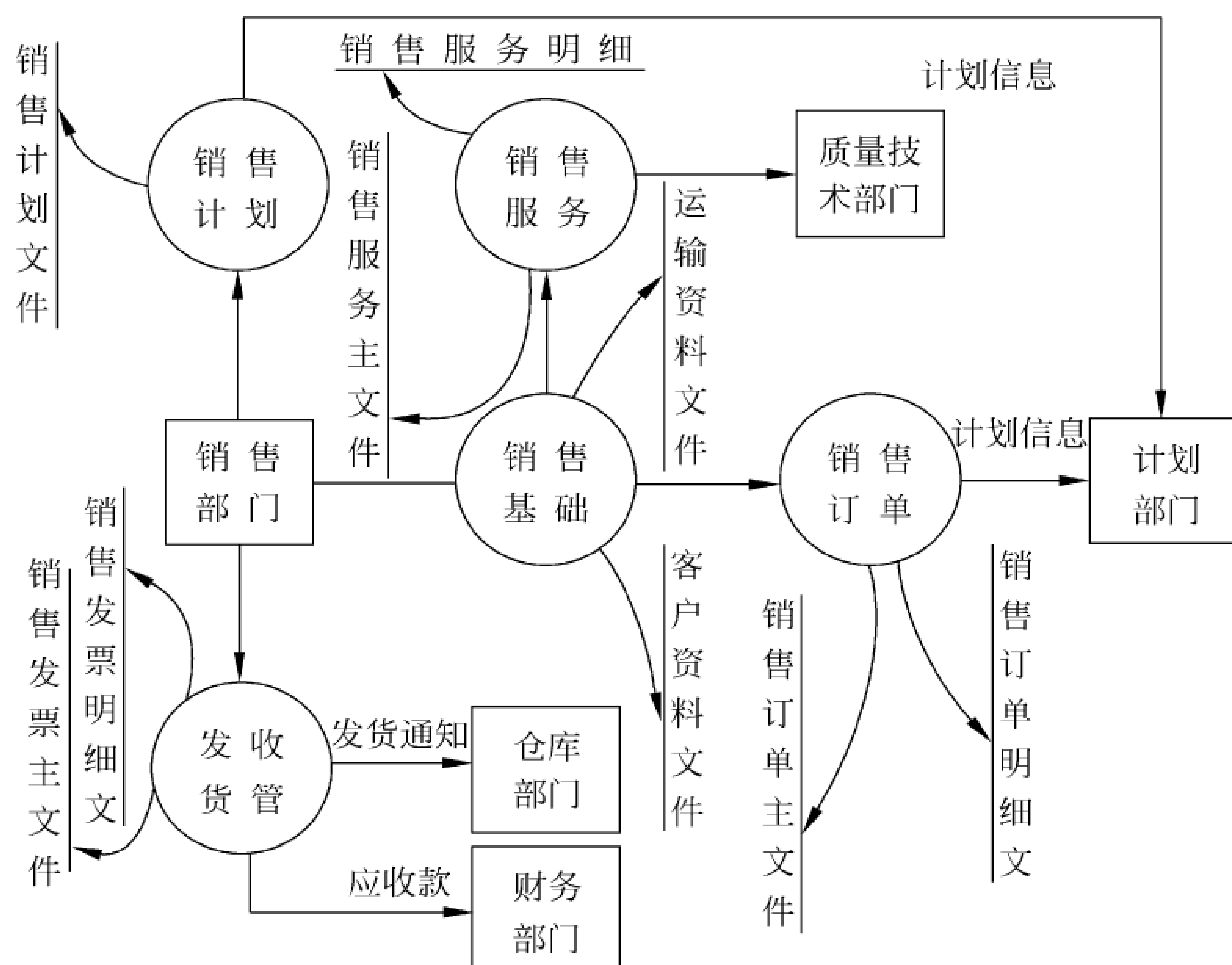


图 8-12　企业销售管理业务的第一层数据流图

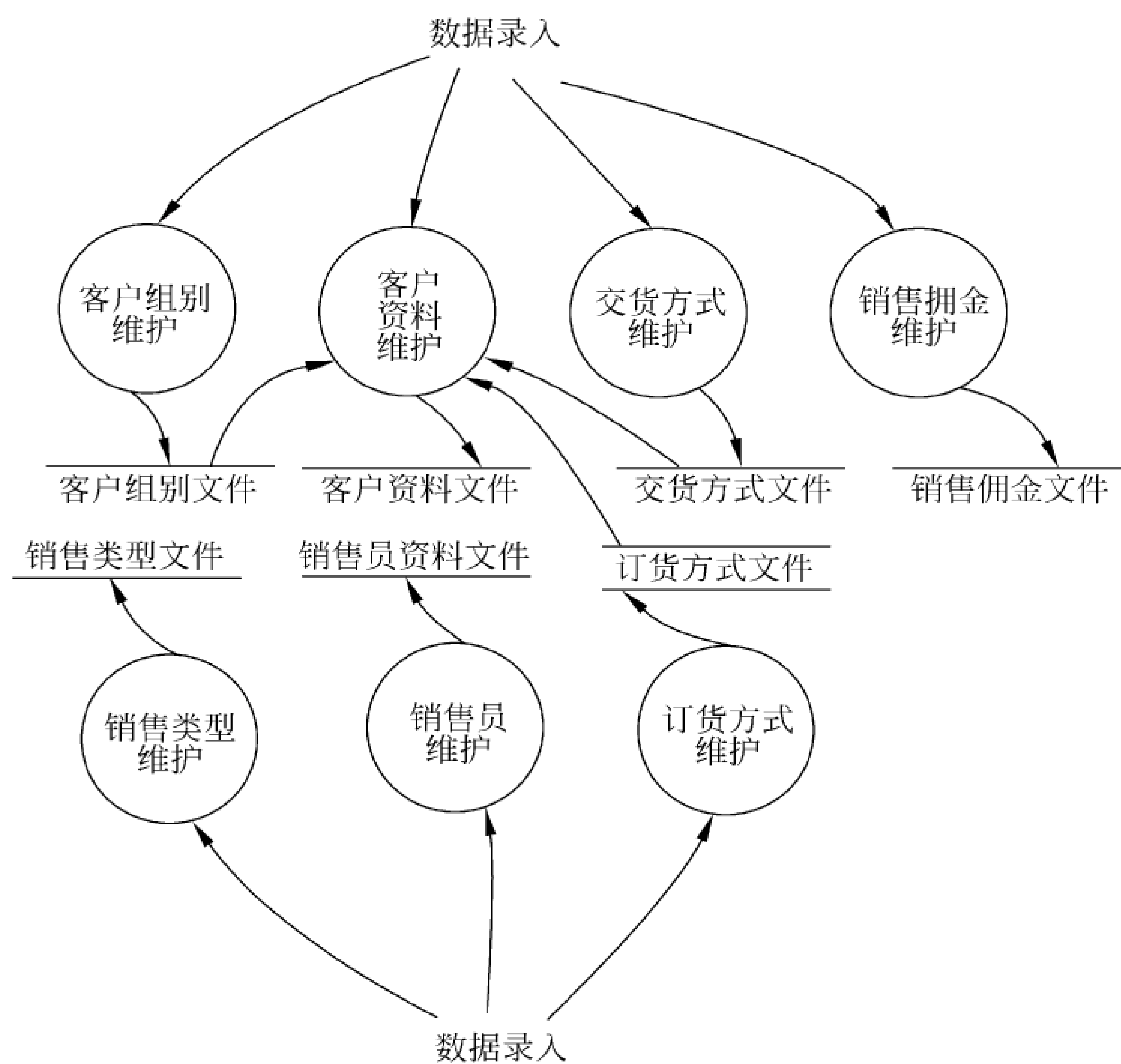


图 8-13　销售基础数据管理业务数据流图（第二层数据流）

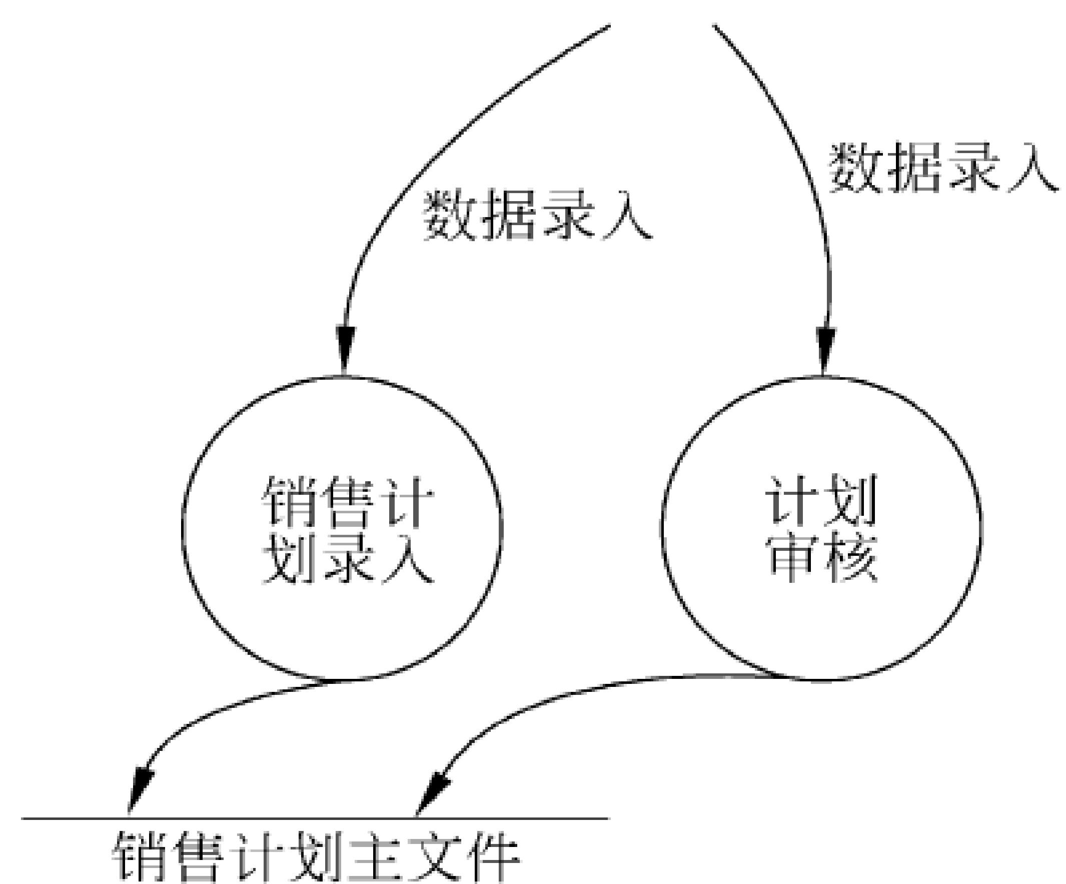


图 8-14　销售计划管理业务数据流图（第二层数据流）

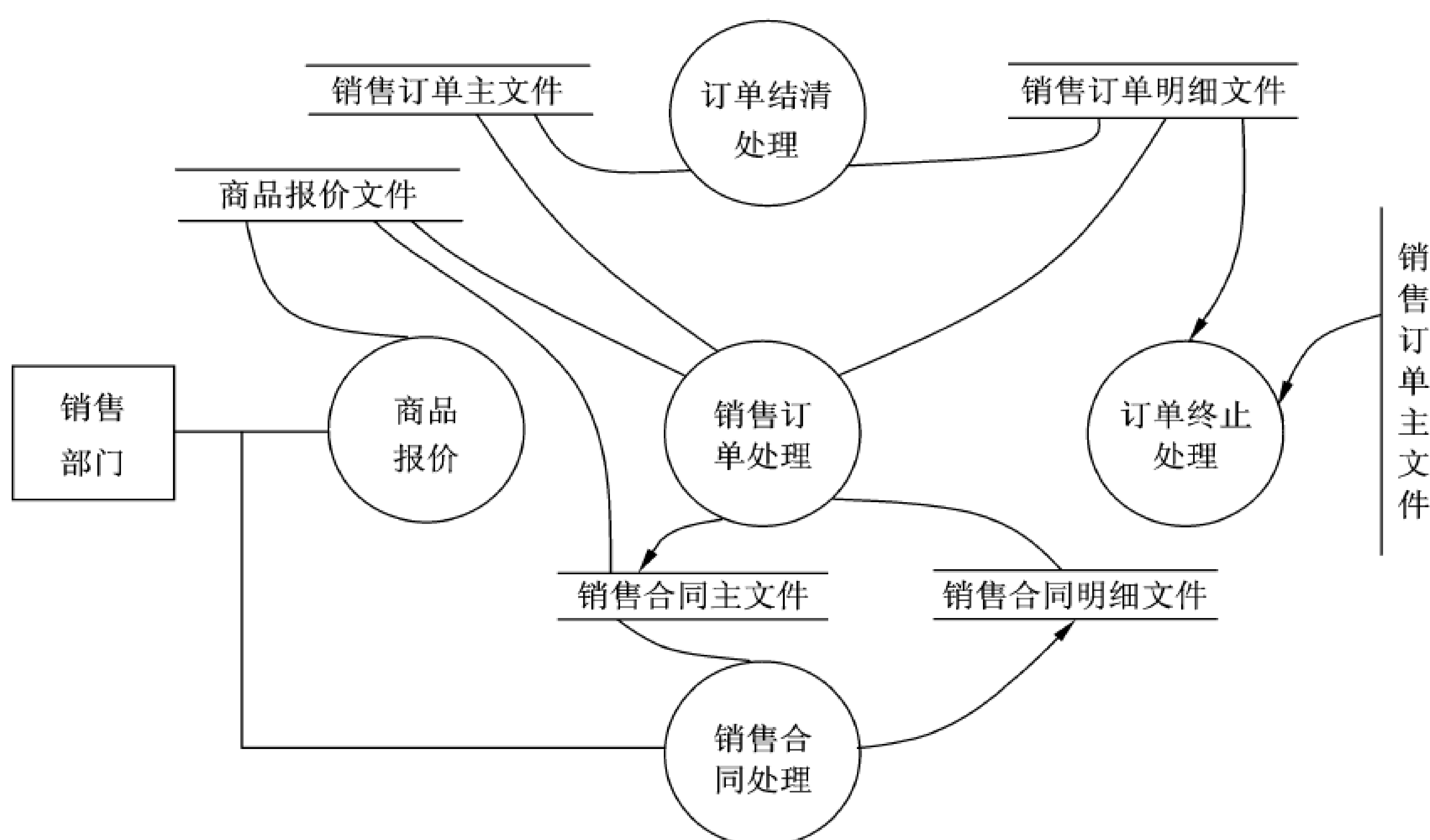


图 8-15　销售订单管理业务数据流图（第二层数据流）

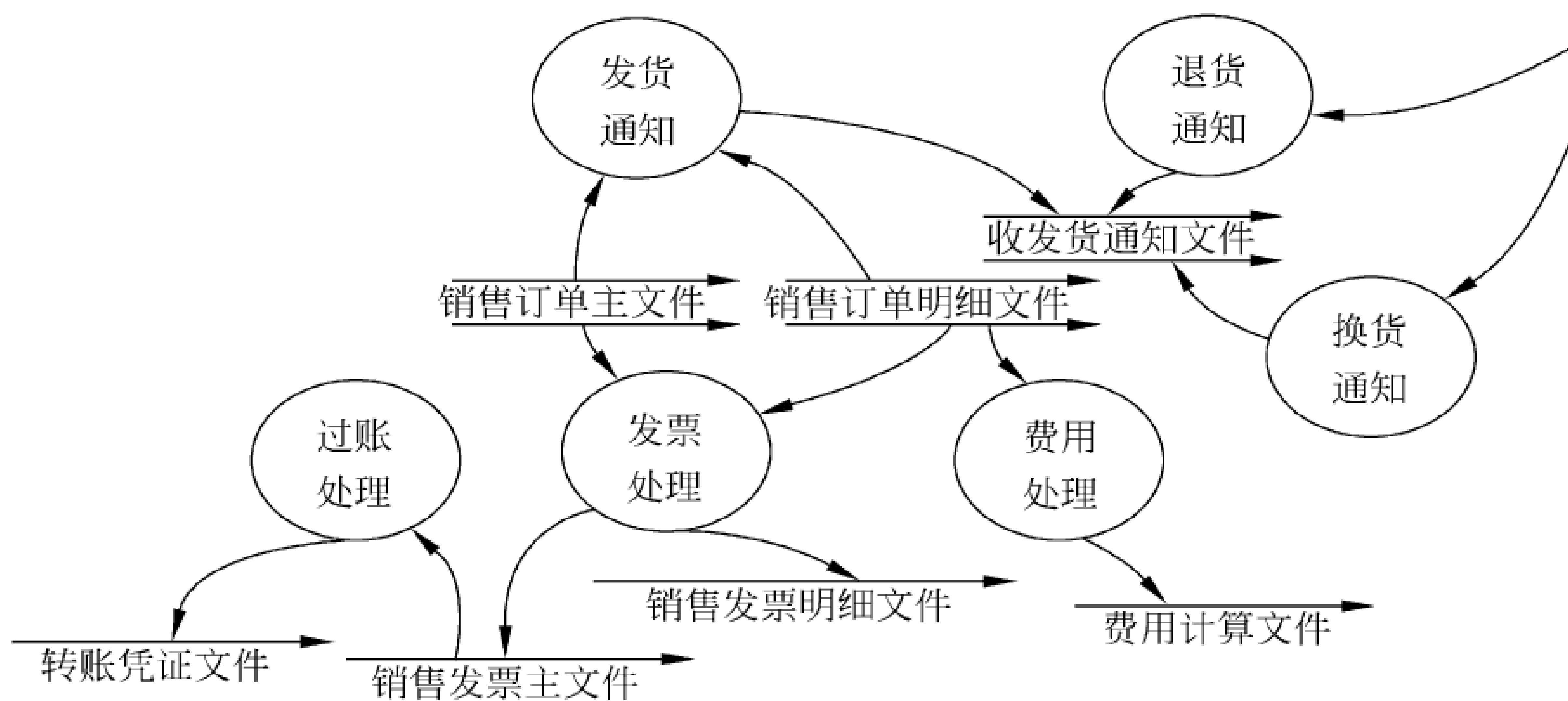


图 8-16　销售收发货管理业务数据流图（第二层数据流）

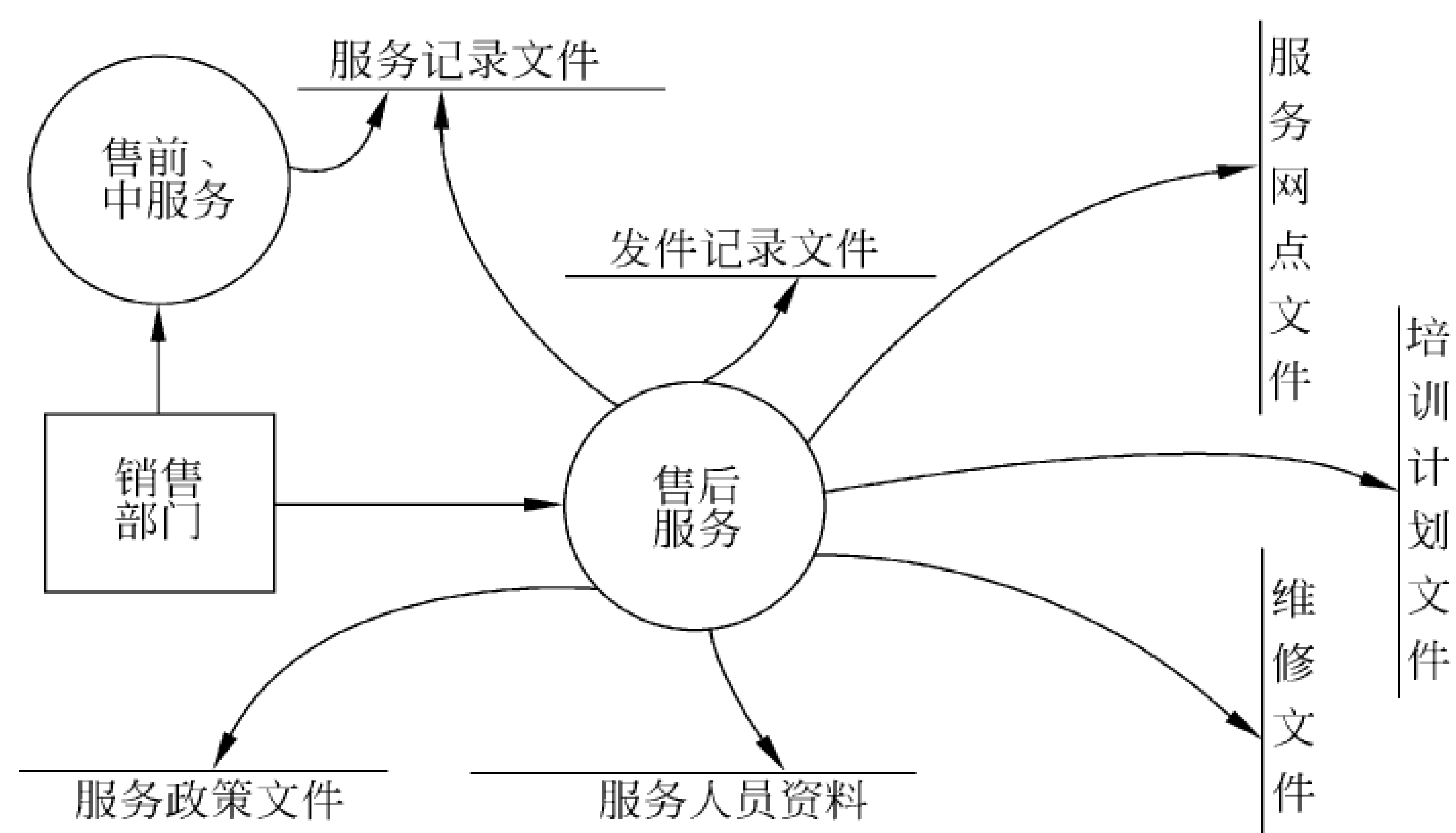


图 8-17　销售服务管理业务数据流图（第二层数据流）

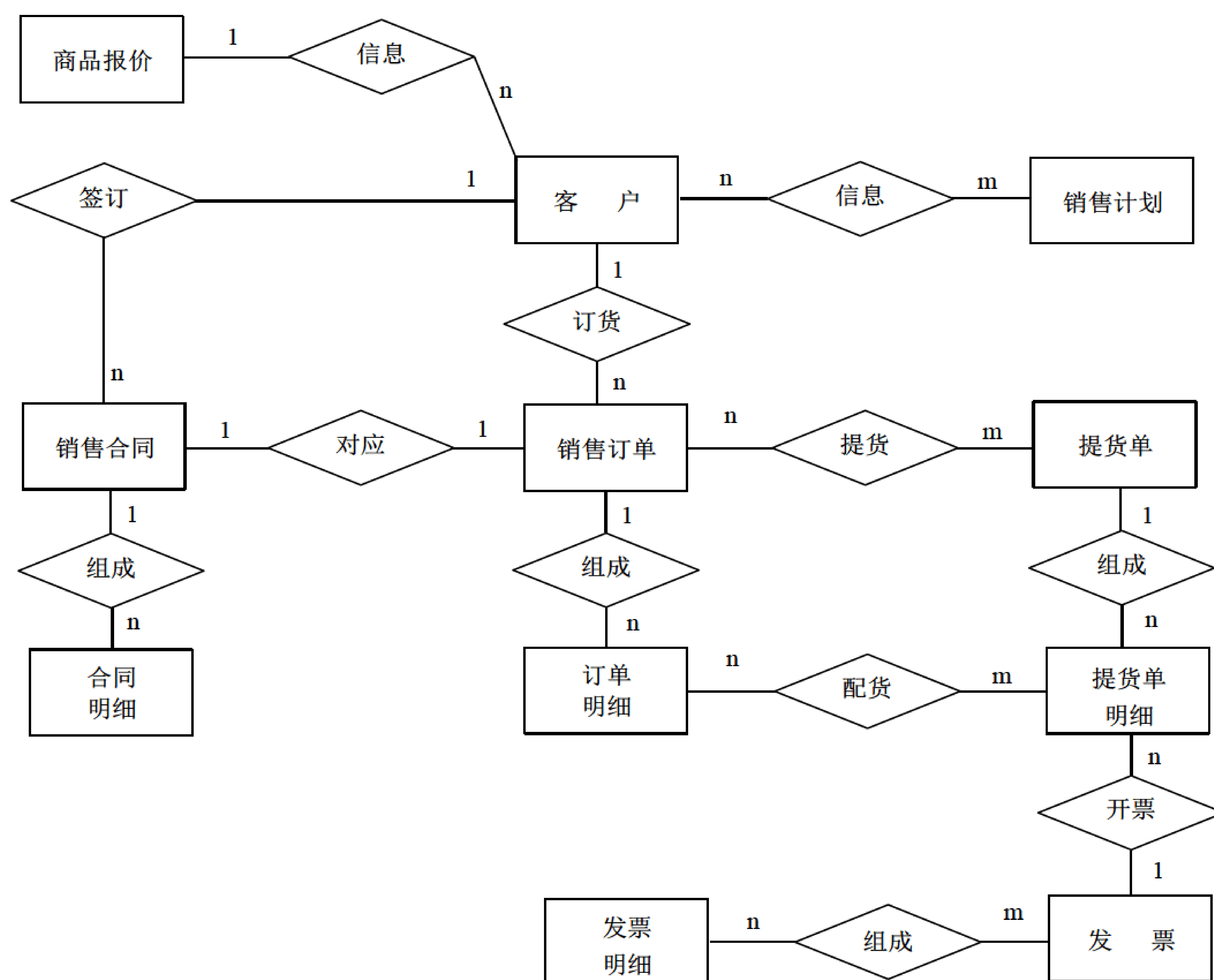


图 8-18　企业销售管理 E-R 关系图

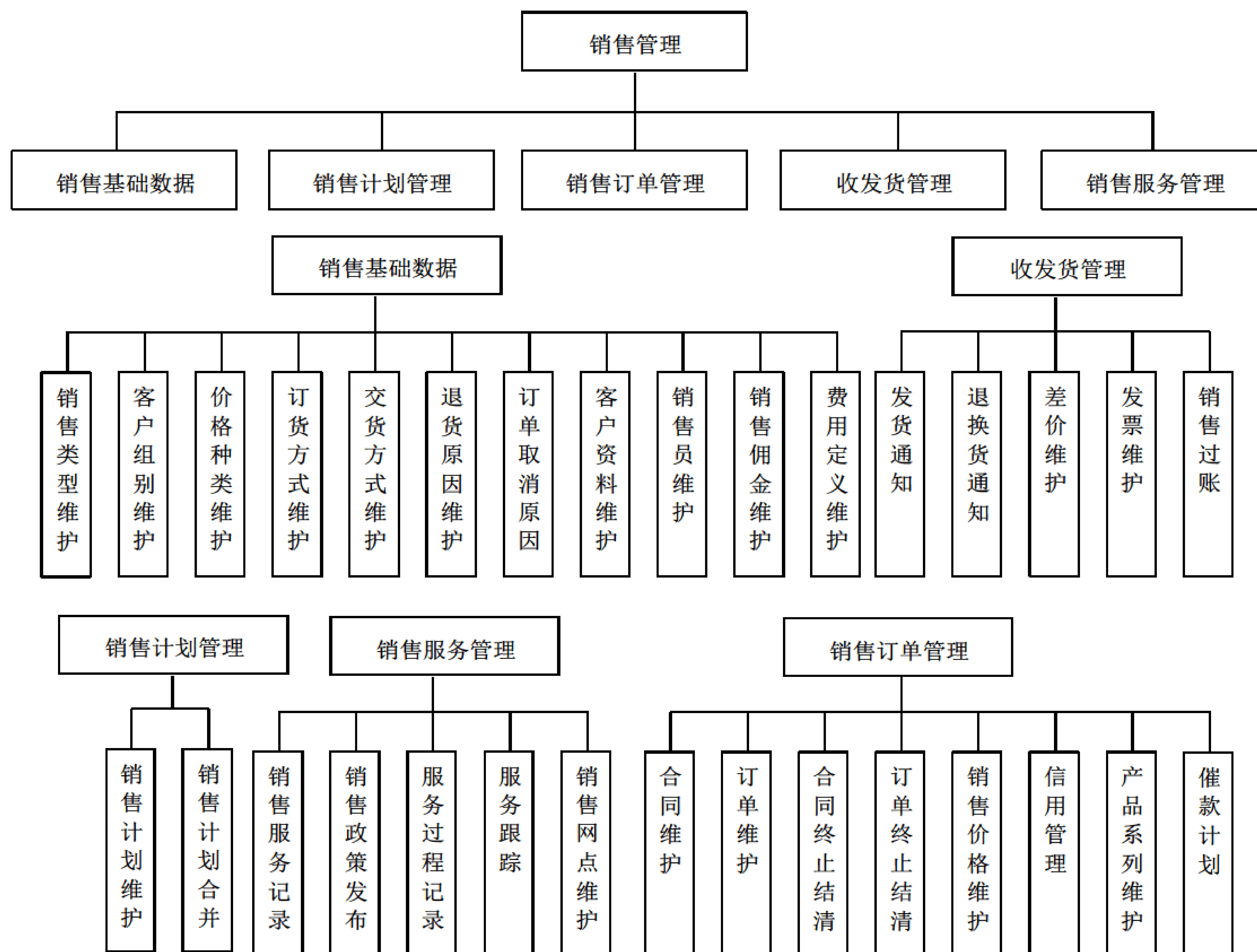


图 8-19　销售管理系统的功能模块图

8.3　决策支持系统的建立

早期的管理信息系统是一个将计算机应用于一个单位或部门的各种业务处理的系统。它将数据处理和经济管理结合起来，形成了用于管理的信息系统。它的定义可以认为是：管理信息系统是一个由人、计算机结合的对管理信息进行收集、传递、存储、加工、维护和使用的系统。管理信息系统能够使各企事业单位的管理由原来的人工处理大量繁琐事务变成了计算机的科学管理，从而使管理提高到新的水平。管理信息系统的最大优点是它能对大量的数据进行有效的管理和处理。它的局限在于对管理者提供的辅助决策信息只表现为数据的查询和统计形式。此时决策支持系统（Decision Support System，DSS）在管理信息系统的基础上发展起来的，最早于 20 世纪 70 年代初由美国 M.S.Scott Morton 教授在《管理决策系统》一文中首先提出。图 8-20 给出了本节内容的知识框图。

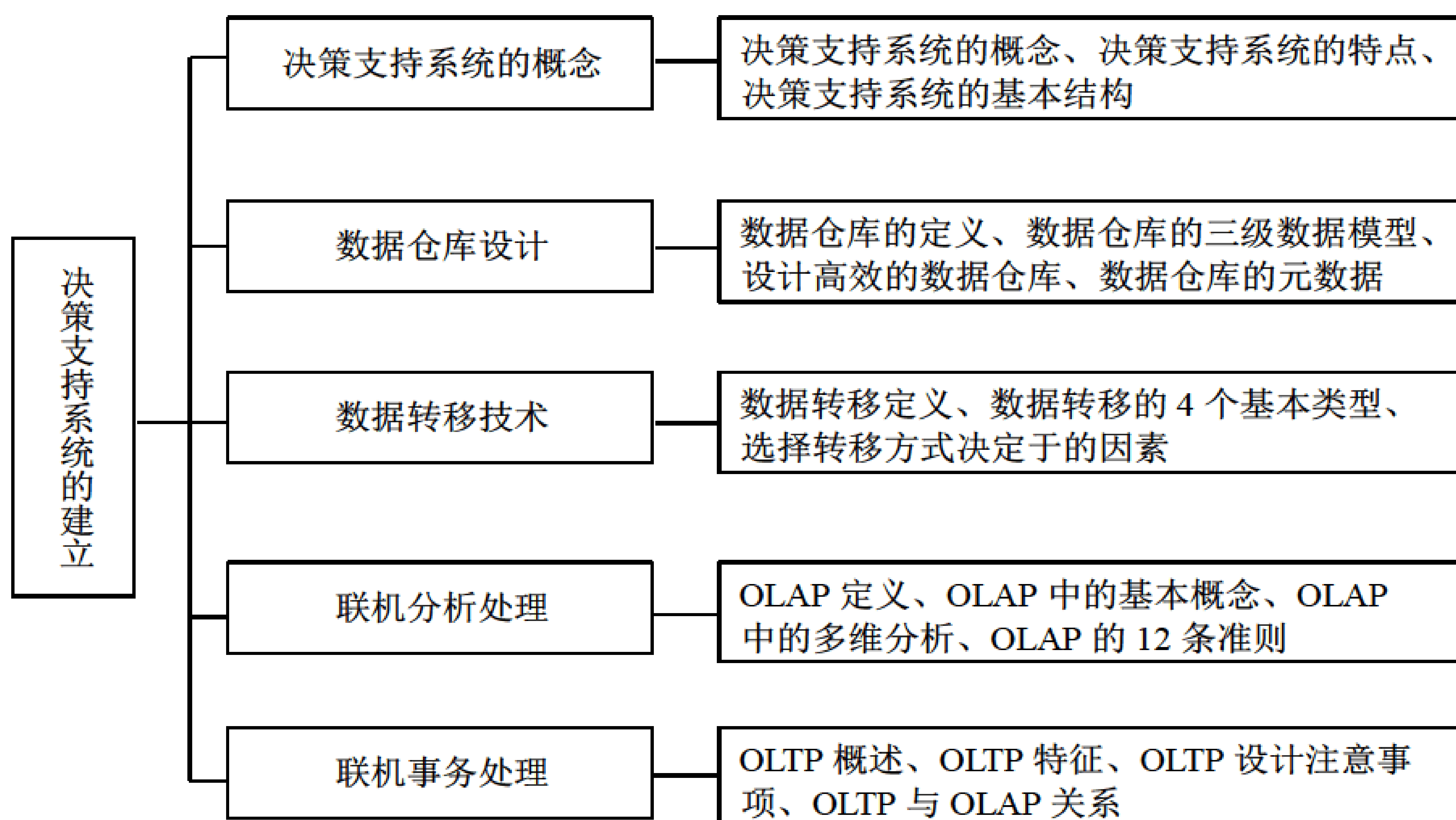


图 8-20　决策支持系统的建立知识框图

8.3.1　决策支持系统的概念

1．知识点提炼

（1）决策支持系统的概念

DSS 是在管理信息系统和运筹学的基础上发展起来的。管理信息系统重点在于对大量数据的处理。运筹学在运用模型辅助决策，体现在单模型辅助决策上。随着新技术的发展，所需要解决的问题会越来越复杂，所涉及的模型越来越多，模型种类也由数学模型再扩充数据处理模型，用来解决一个大问题的模型数量可能成百上千。这样，对多模型辅助决策问题，在决策支持系统出现之前是靠人来实现模型间的联合和协调。决策支持系统的出现是要解决由计算机自动组织和协调多模型的运行，对大量数据库中数据的存取和处理，达到更高层次的辅助决策能力。

（2）决策支持系统的特点

决策支持系统的新特点是增加了模型库和模型库管理系统，它把众多的模型（数学模型与数据处理模型以及更广泛的模型）进行有效组织和存储，并且建立了模型库和数据库的有机结合。这种有机结合适应人机交互功能，自然促使新型系统的出现，即 DSS 的出现。它不同于 MIS 数据处理，也不同于模型的数值计算，而是它们的有机集成。它既具有数据处理功能又具有数值计算功能。

2．难点分析

决策支持系统由图 8-21 所示的子系统组成。

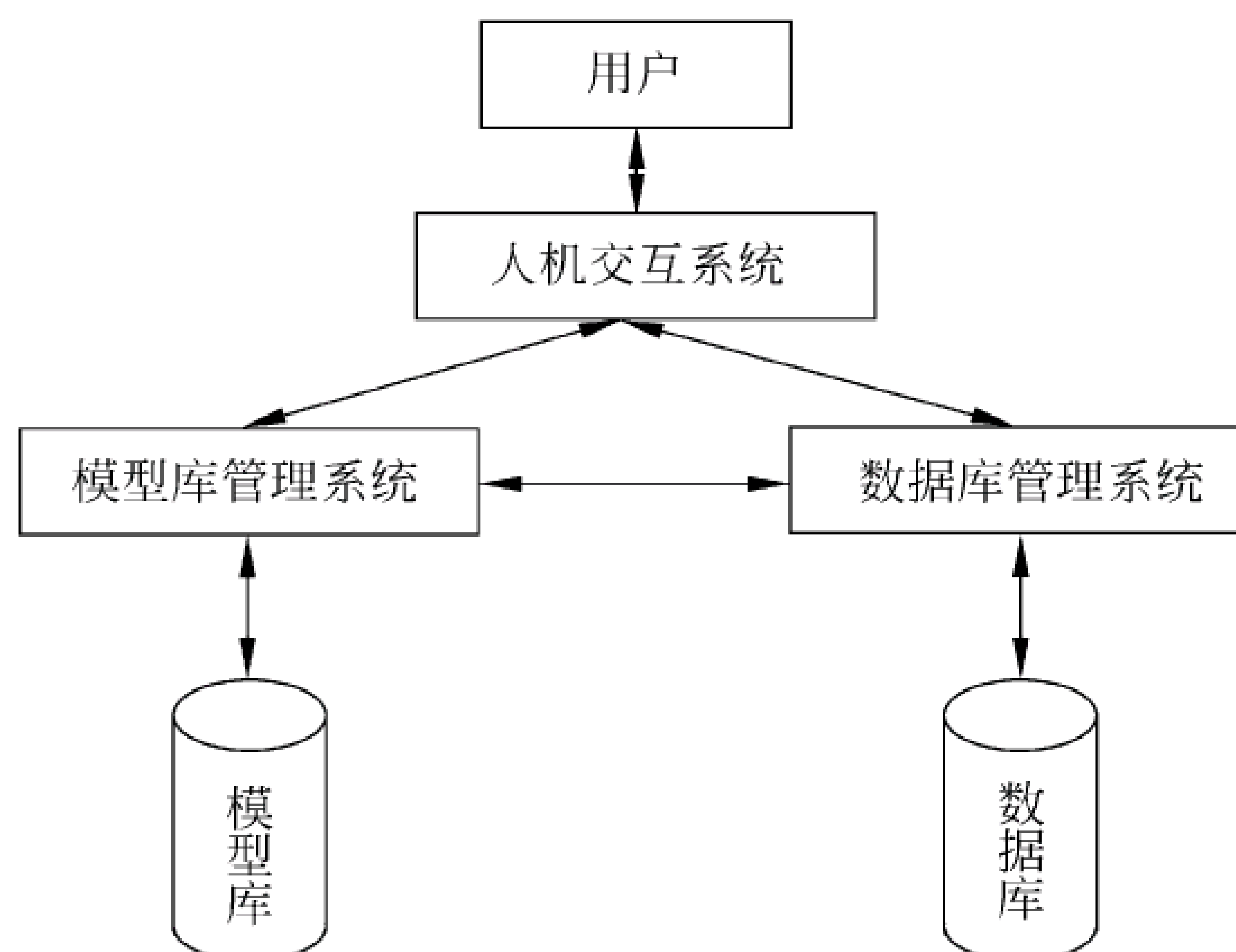


图 8-21　决策支持系统的构成

① 数据库子系统。包括数据库，其中包含关于决策问题的有关数据，并由数据库管理系统进行管理。

② 模型库子系统。包括模型库，其中包含财务、统计、管理科学或其他定量模型，可提供系统的分析功能，由模型库管理系统为用户提供建模语言以及模型库管理功能。

③ 人机交互系统。通过该子系统，用户与决策支持系统通信并使用决策支持系统，协调和控制数据库子系统和模型库子系统的管理和运行。

④ 用户。用户也是系统的一部分，研究人员认为用户与计算机的频繁对话可以产生决策支持系统某些特殊的作用。

⑤ 扩展。用户在决策支持系统基本结构的基础上，增加知识库子系统，就形成了智能决策支持系统结构。该结构中的知识库子系统包括知识库和推理机。知识库子系统能支持其他子系统或作为独立的部件应用，它提供智能和定性分析功能，以增强决策人的能力。

3．典型例题

【例题 8-8】 决策支持系统的新特点是增加了__①__和__②__，建立了__①__和数据库的有机结合。这种有机结合适应人机交互功能，自然促使新型系统的出现，即 DSS 的出现。它不同于 MIS 数据处理，也不同于模型的数值计算，而是它们的有机集成。它既具有数据处理功能又具有数值计算功能。

① A．知识库　　B．模型库
C．数据模型　　D．数据处理模型

② A．数据库管理系统　　B．知识库管理系统
C．决策文件管理系统　　D．模型库管理系统

【答案】 ① B　② D

【解析】 决策支持系统的新特点是增加了模型库和模型库管理系统，它把众多的模型（数学模型与数据处理模型以及更广泛的模型）进行有效组织和存储，并且建立了模型库和

数据库的有机结合。这种有机结合适应人机交互功能，自然促使新型系统的出现，即 DSS 的出现。它不同于 MIS 数据处理，也不同于模型的数值计算，而是它们的有机集成。它既具有数据处理功能又具有数值计算功能。

8.3.2　数据仓库设计

由于数据库面向日常事务处理，不适合进行分析处理，一种新的技术应运而生，这就是数据仓库技术。数据仓库技术是公认的信息利用的最佳解决方案，它不仅能够从容解决信息技术人员面临的问题，同时也为商业用户提供很好的商业契机。

1．知识点提炼

（1）数据仓库的定义

数据仓库（Data Warehouse）是在 20 世纪 80 年代中期，由“数据仓库之父”William H.Inmon 先生在其《建立数据仓库》一书中定义了数据仓库的概念，随后又给出了更为精确的定义：数据仓库是在企业管理和决策中面向主题的、集成的、与时间相关的、不可修改的数据集合。

（2）数据仓库三级数据模型

① 概念模型

概念模型常用的表示方法是使用 E-R 图作为它的描述工具。E-R 图描述的是实体以及实体之间的联系，在 E-R 图中，长方形表示实体，在数据仓库中就表示主题，在长方形内写上主题名；椭圆形表示主题的属性，并用无向边把主题与其属性连接起来；用菱形表示主题之间的联系，菱形框内写上联系的名字。用无向边把菱形分别与有关的主题连接，给无向边标记上联系的类型。若主题之间的联系也具有属性，则把属性和菱形也用无向边连接上。

由于 E-R 图具有良好的可操作性，形式简单，易于理解，便于与用户交流，对客观世界的描述能力也较强，所以在数据库设计方面更得到了广泛的应用。因为目前的数据仓库一般建立在关系数据库的基础之上，为了和原有数据库的概念模型相一致，采用 E-R 图作为数据仓库的概念模型仍然是较为合适的。

② 逻辑模型

在数据仓库的设计中采用的逻辑模型就是关系模型。无论是主题还是主题之间的联系，都用关系来标识。关系模型概念简单、清晰，用户易懂、易用，有严格的数学基础和在此基础上发展的关系数据理论；关系模型简化了程序员的工作和数据仓库设计开发的工作，当前比较成熟的商品化数据库产品都是基于关系模型的。因此采用关系模型作为数据仓库的逻辑模型是合适的。数据仓库的逻辑模型描述了数据仓库主题的逻辑实现，即每个主题所对应的关系表的关系模式的定义。

③ 物理模型

所谓数据仓库的物理模型就是逻辑模型在数据仓库中的实现，如物理存取方式、数据

存储结构、数据存放位置以及存储分配等等。物理模型是在逻辑模型的基础之上实现的，在进行物理模型设计实现时，所考虑的主要因素有：I/O 存取时间、空间利用率和维护代价；在进行数据仓库的物理模型设计时，考虑到数据仓库的数据量大，但是操作单一的特点，可采取其他的一些提高数据仓库性能的技术，如合并表、建立数据序列、引入冗余、进一步细分数据、生成导出数据、建立广义索引等等。

（3）数据仓库的数据模型与操作型数据库的三级数据模型的区别

数据仓库的数据模型与操作型数据库的三级数据模型的区别主要表现在以下几个方面。

① 数据仓库的数据模型中不包含纯操作型的数据。

② 数据仓库的数据模型扩充了码结构，增加了时间属性作为码的一部分。

③ 数据仓库的数据模型中增加了一些导出数据。

可以看出，上述 3 点差别也就是操作型环境中的数据与数据仓库中的数据之间的差别，同样是数据仓库为面向数据分析处理所要求的。虽然存在着这样的差别，在数据仓库设计中，仍然存在着三级数据模型，即概念模型、逻辑模型和物理模型。

（4）设计高效的数据仓库

建立数据仓库过程中一个重要问题是如何提高系统的性能。因为数据仓库的数据量很大，分析处理时涉及的数据范围也较广，往往涉及大规模数据的查询。提高系统性能，主要是要提高系统的物理 I/O 性能，因为 I/O 瓶颈常成为影响系统性能的主要因素。在数据仓库的设计中，应尽量减少每次查询处理要求的 I/O 次数，而使每次 I/O 又能返回尽量多的记录。事实上，由于数据仓库的数据极少甚至不再更新，数据仓库的物理设计可以有更多的方法和途径来提高系统性能。但最基本的方法和原则还是下面介绍的粒度划分和数据分割。

① 粒度划分

对数据仓库开发者来说，划分粒度是设计过程中最重要的问题之一。所谓粒度是指数据仓库中数据单元的详细程度和级别。数据越详细，粒度越小级别就越低；数据综合度越高，粒度越大级别就越高。在传统的操作型系统中，对数据的处理和操作都是在详细数据级别上的，即最低级的粒度。但是在数据仓库环境中主要是分析型处理，粒度的划分将直接影响数据仓库中的数据量以及所适合的查询类型。一般需要将数据划分为：详细数据、轻度总结、高度总结三级或更多级粒度。不同粒度级别的数据用于不同类型的分析处理。粒度的划分是数据仓库设计工作的一项重要内容，粒度划分是否适当是影响数据仓库性能的一个重要方面。

② 数据分割

数据分割是数据仓库设计的另一项重要内容，是提高数据仓库性能的一项重要技术。数据的分割是指把逻辑上是统一整体的数据分割成较小的、可以独立管理的物理单元进行存储，以便于重构、重组和恢复，以提高创建索引和顺序扫描的效率。数据的分割使数据仓库的开发人员和用户具有更大的灵活性。数据仓库中数据分割的概念与数据库中的数据

分片概念是相近的。数据库系统中的数据分片有水平分片、垂直分片、混合分片和导出分片多种方式。水平分片是指按一定的条件将一个关系按行分为若干不相交的子集，每个子集为关系的一个片段；垂直分片是指将关系按列分为若干子集，垂直分片的片段必须能够重构原来的全局关系。

在作数据仓库设计时需要把数据分割与粒度划分综合起来考虑，并在设计之前做好业务规划，将数据访问频率、元数据大小等因素纳入业务规划的思考范畴。

（5）数据仓库的元数据

数据仓库中的元数据就是关于数据的数据，它描述了数据的结构、内容、码、索引等项内容。传统数据库中的数据字典是一种元数据，但在数据仓库中，元数据的内容比数据库中的数据字典更丰富、更复杂。设计一个描述能力强、内容完善的元数据，是有效管理数据仓库、具有决定意义的重要前提。因此元数据的设计在整个数据仓库设计中占有重要的地位，是数据仓库设计的一个重要组成部分。

2．难点分析

数据仓库中的元数据的重要性表现在以下几个方面。

① 数据仓库服务于决策支持系统分析员以及高层决策人员，而这一部分人员往往把使用元数据作为分析的第一步。例如，数据仓库元数据中的广义索引中存有在每次数据装载时产生的部分有关决策的数据，在做决策时，可以先去查找这部分数据，再决定是否进行进一步的搜索。

② 操作型环境和数据仓库环境之间有着复杂的、多方面的区别，因此从操作型环境到数据仓库的数据转换也是复杂的、多方面的。元数据应包含对这种转换的描述。元数据要将这种转换清晰地表示出来，把从哪些数据源用怎样的转换逻辑转换成数据仓库中的哪些目的数据等内容描述出来。这样，当从数据仓库向数据库回溯时，便能够根据数据变换的历史，找到原始依据。数据仓库的元数据还要将这种转换管理起来，既保证这种转换是正确的、适当的或合理的，又要使其是可变的、灵活的。事实上，因为用户需求是不确定的，只有保证元数据的灵活性、可变性，才能真正保证其合理性和正确性。

③ 除了描述和管理从数据库到数据仓库的转换外，数据仓库的元数据当然还要管理好数据仓库中的数据。一方面，数据仓库中的数据量很大，划分不同的粒度层次、进行分割策略的选择、建立各种各样的索引等等，都需要在元数据中进行描述和管理；另一方面，数据仓库中包含着较长时期内的数据，不同时期不同的需求使得其数据从形式到内容都可能不同。

元数据的内容在数据仓库设计、开发、实施，以及使用过程中不断完善，不仅为数据仓库的创建提供必要的信息、描述和定义，还为决策支持系统分析人员访问数据仓库提供直接的或辅助的信息。

3．典型例题

【例题 8-9】 建立数据仓库过程中一个重要问题是如何提高系统的性能。其物理设计

可以有更多的方法和途径来提高系统性能。但最基本的方法和原则还是 ① 和 ② 。

① A．内外模式划分 B．存储空间划分 C．粒度划分 D．权限划分

② A．数据分割 B．数据集中 C．数据整合 D．数据分类

【答案】 ① C ② A

【解析】 数据仓库的物理设计可以有更多的方法和途径来提高系统性能。但最基本的方法和原则还是下面介绍的粒度划分和数据分割。

① 粒度划分

对数据仓库开发者来说，划分粒度是设计过程中最重要的问题之一。所谓粒度是指数据仓库中数据单元的详细程度和级别。数据越详细，粒度越小级别就越低；数据综合度越高，粒度越大级别就越高。在传统的操作型系统中，对数据的处理和操作都是在详细数据级别上的，即最低级的粒度。但是在数据仓库环境中主要是分析型处理，粒度的划分将直接影响数据仓库中的数据量以及所适合的查询类型。一般需要将数据划分为：详细数据、轻度总结、高度总结三级或更多级粒度。不同粒度级别的数据用于不同类型的分析处理。粒度的划分是数据仓库设计工作的一项重要内容，粒度划分是否适当是影响数据仓库性能的一个重要方面。

② 数据分割

数据分割是数据仓库设计的另一项重要内容，是提高数据仓库性能的一项重要技术。数据的分割是指把逻辑上是统一整体的数据分割成较小的、可以独立管理的物理单元进行存储，以便于重构、重组和恢复，以提高创建索引和顺序扫描的效率。数据的分割使数据仓库的开发人员和用户具有更大的灵活性。数据仓库中数据分割的概念与数据库中的数据分片概念是相近的。数据库系统中的数据分片有水平分片、垂直分片、混合分片和导出分片多种方式。水平分片是指按一定的条件将一个关系按行分为若干不相交的子集，每个子集为关系的一个片段；垂直分片是指将关系按列分为若干子集，垂直分片的片段必须能够重构原来的全局关系。

在作数据仓库设计时需要把数据分割与粒度划分综合起来考虑，并在设计之前做好业务规划，将数据访问频率、元数据大小等因素纳入业务规划的思考范畴。

8.3.3 数据转移技术

数据仓库的基本观念之一是，当数据从业务系统或其他数据来源提取出来时，应该先经过变换或清洗，才能将它加载到数据仓库中。然而，对于数据转移的目的和实现转移的最优方法却存在很多混乱的看法。在数据仓库环境中进行数据转移的目的应该有两个：第一是改进数据仓库中数据的质量；第二是提高数据仓库中数据的可用性。

1．知识点提炼

（1）数据转移定义

所谓数据转移，也称为数据转换或数据变换，就是把多种传统资源或外部资源信息中

不完善的数据自动转换为商务中准确可靠的数据。为了便于讨论，将把业务数据加载到数据仓库之前经历的内存和结构的变化都纳入数据转移。

在建立数据仓库的过程中，数据转移是个重要的步骤。实施数据转移的好方法有很多，既有定制代码的程序，也有用于转移仓库数据的专门工具。无论采用什么方法，转移的几个基本方面是必须实现的。转移远远不止是把数据移入数据仓库时改变它的数据结构，真正好的转移是数据进入数据仓库时能够检验并提高它的质量和可用性。

（2）数据转移的 4 个基本类型

为了对数据转移的复杂性进行深入的讨论，必须定义数据转移的几个基本类型，每一类都有自己的特点和表现形式。为了便于讨论，应该考虑以下 4 种转移类型。

① 简单转移。简单转移是所有数据转移的基本构成单元。这一类中包括的数据处理一次只针对一个字段，而不考虑相关字段的值。

② 清洗。清洗的目的是为了保证前后一致地格式化和使用某一字段或相关的字段群。例如，它可以包括地址信息的适当格式化。清洗还能检查某一特定字段的有效值，通常是通过进行范围检查或是从枚举清单中做出选择。

③ 集成。集成是指将业务数据从一个或几个来源中取出，并逐字段地将数据映射到数据仓库的新数据结构上。

④ 聚集和概括。聚集和概括是把业务环境中找到的零星数据压缩成仓库环境中的较少数据块。

2．难点分析

下面介绍的是选择转移方式取决的因素。

① 时间范围。虽然使用数据转移工具能大大方便建立和维护数据仓库的过程，但获取、配置和学习这些工具都要花些时间。如果生产第一个仓库应交付成果的时间约束很紧，那么这样的项目往往选择人工编码，在仓库建设得较完善、项目小组的可信度较高时再迁移到数据转移工具中。

② 预算。数据仓库工具可能会很贵，但编程人员编写变换程序的时间也一样宝贵，因此最佳选择取决于哪类预算近期在组织中更容易得到。但从长期来看，投资可以比手工编制和维护变换程序的成本要节省得多。

③ 数据仓库的规模。范围很小的数据仓库（即数据源很少，要实现的转移也很少）可能无法证实使用数据转移工具的成本合理性。然而，当数据仓库的范围逐渐扩大时，维护手工编写的变换程序将变得越来越困难，工具就会发挥更大的作用。但对于一个规模很小的初始数据仓库来说，当然就不需要变换工具了。

④ 数据仓库项目小组的规模和技能。如果仓库小组足够大，而且具有适当的编程技巧，建立和维护转移逻辑就容易一些。但是大多数数据仓库小组都受到资源的限制，无法得到足够的开发时间维护所有的变换代码。因此他们发现转移工具特别有用，因为数据仓库分析人员可以不要编程人员的帮助自己建立并维护大多数的转移。

当然，使用数据转移工具的最主要原因之一与节省时间和有效利用成本没有太大的关系。这个原因是好的数据转移工具能自动地生成并维护宝贵的元数据。

3．典型例题

【例题 8-10】 数据仓库的基本观念之一是当数据从业务系统或其他数据来源提取出来时，应该先经过变换或清洗，才能将它加载到数据仓库中。数据转移有几种基本类型，包括简单转移、__①__、集成、__②__。

① A．分类　　B．清洗
　C．定义数据项　　D．更改数据存储方式

② A．聚集和概括　　B．聚集或分类
　C．分类和分项存储　　D．减少或增加数项

【答案】 ① B　② A

【解析】 数据转移的 4 个基本类型，每一类都有自己的特点和表现形式。

① 简单转移。简单转移是所有数据转移的基本构成单元。这一类中包括的数据处理一次只针对一个字段，而不考虑相关字段的值。

② 清洗。清洗的目的是为了保证前后一致地格式化和使用某一字段或相关的字段群。例如，它可以包括地址信息的适当格式化。清洗还能检查某一特定字段的有效值，通常是通过进行范围检查或是从枚举清单中做出选择。

③ 集成。集成是指将业务数据从一个或几个来源中取出，并逐字段地将数据映射到数据仓库的新数据结构上。

④ 聚集和概括。聚集和概括是把业务环境中找到的零星数据压缩成仓库环境中的较少数据块。

8.3.4 联机分析处理

20 世纪 60 年代末，E.F.Codd 所提出的关系数据模型促进了关系数据库及联机事务处理（On-Line Transaction Processing，OLTP）的发展。数据不再以文件方式同应用程序捆绑在一起，而是分离出来以关系表方式供大家共享。随着政府及商业应用的发展，数据量越来越大，同时用户的查询需求也越来越复杂，涉及的已不仅是查询或操纵一张关系表中的一条或几条记录，而是要对多张表中千万条记录进行数据分析和信息综合。关系数据库系统已不能全部满足这一要求。这两类应用，操作型应用和分析型应用，特别是在性能上难以两全，尽管为了提高性能，人们常常在关系数据库中放宽了对冗余的限制，引入了统计及综合数据，但这些统计综合数据的应用逻辑却是分散杂乱的，非系统化的，因此分析功能有限、不灵活，维护困难。在国外，不少软件厂商采取了发展其前端产品来弥补关系数据库管理系统支持的不足，他们通过专门的数据综合引擎，辅之以更加直观的数据访问界面，力图统一分散的公共应用逻辑，在短时间内响应非数据处理专业人员的复杂查询要求。1993 年，E.F.Codd 将这类技术定义为联机分析处理（On-Line Analytical Processing，OLAP）。

OLAP 作为一类产品同 OLTP 明显区分开来。

1. 知识点提炼

（1）OLAP 定义

OLAP 是针对待定问题的联机数据访问和分析。为了反映用户所能理解的企业真实的“维”，原始的数据被进行了转换，从而形成了可用的信息。通过对信息的很多种可能的观察形式进行快速、稳定一致的交互性存取，允许管理决策人员对数据进行深入观察。

OLAP 是以数据仓库进行分析决策的基础，针对特定问题的联机数据访问和分析，OLAP 能够对不同数据集合进行基于某个或是多个角度的比较，它能够从不同角度切割数据集合从而进行分析。从某种意义来说，OLAP 是有预见性的。OLAP 的分析是建立在经验的基础上，对数据进行某种指定关联的分析。在联机事务处理系统中，由于数据的离散性，而使 OLAP 实现起来相当复杂甚至是不可能，而以数据仓库为依托，辅之以 OLAP 工具，OLAP 的实现将十分简单易行。

（2）OLAP 中的基本概念

① 变量：变量是数据的实际意义，即描述数据“是什么”。例如数据 10 000 本身并没有意义或者意义未定，它可能是一个学校的学生人数，也可能是某产品的单价，还可能是某商品的销售量等等。一般情况下，变量总是一个数值度量指标，例如“人数”、“单价”、“销售量”等都是变量，而 10 000 则是变量的一个值。

② 维：维是人们观察数据的特定角度。例如，企业常常关心产品销售数据随着时间推移而产生的变化情况，这时是从时间的角度来观察产品的销售，所以时间就是一个维，简称为时间维。企业也时常关心自己的产品在不同地区的销售分布情况，这时是从地理分布的角度来观察产品的销售，所以地理分布也是一个维，称为地理维。

③ 维的层次：人们观察数据的某个特定角度（即某个维）还可以存在细节程度不同的多个描述方面，称这多个描述方面为维的层次。一个维往往具有多个层次，例如描述时间维，可以从日期、月份、季度、年等不同层次来描述，那么日期、月份、季度、年等就是时间维的层次；同样，城市、地区、国家等构成了一个地理维的多个层次。

④ 维成员：维的一个取值称为该维的一个维成员。如果一个维是多层次的，那么该维的维成员是在不同维层次的取值的组合。例如，考虑时间维具有日期、月份、年这 3 个层次，分别在日期、月份、年上各取一个值组合起来，就得到了时间维的一个维成员，即“某年某月某日”。一个维成员并不一定在每个维层次上都要取值，例如，“某年某月”、“某月某日”、“某年”等等都是时间维的维成员。对应一个数据项来说，维成员是该数据项在某维中位置的描述。例如对一个销售数据来说，时间维的维成员“某年某月某日”就表示该销售数据是“某年某月某日”的销售数据，“某年某月某日”是该销售数据在时间维上位置的描述。

⑤ 多维数组：一个多维数组可以表示为：（维 1，维 2，……，维 n，变量）。例如，（地区，时间，销售渠道，销售额）就是一个多维数组，其中销售额是变量，它定义在地区

维、时间维和销售渠道维这三者的基础上。

⑥ 数据单元：多维数组的取值称为数据单元。当在多维数组的各个维中都选中一个维成员，这些维成员的组合就唯一确定了一个变量的值。那么数据单元就可以表示为：（维 1 的维成员，维 2 的维成员，……，维 n 的维成员，变量的维成员。）

（3）OLAP 中的多维分析

多维分析是指对以多维形式组织起来的数据采取切片、切块、旋转等各种分析动作，以求剖析数据，使最终用户能从多个角度、多侧面地观察数据库中的数据，从而深入地了解包含在数据中的信息、内涵。多维分析方式迎合了人的思维模式，因此减少了混淆并且降低了出现错误解释的可能性。多维分析的基本动作有以下几种。

① 切片：在多维数组的某一维上选定一维成员的动作称为切片，即在多维数组（维 1，维 2，……，维 n，变量）中选一维，并取其一维成员。

② 切块：在多维数组的某一维上选定某一区间的维成员的动作称为切块，即限制多维数组的某一维的取值区间。显然，当这一区间只取一个维成员时，即得到一个切片。

③ 旋转：旋转即是改变一个报告或页面显示的维方向。例如，旋转可能包含了交换行和列，或是把某一个行维移到列维中去，或是把页面显示中的一个维和页面外的维进行交换（令其成为新的行或列中的一个）。

OLAP 的数据来源于数据库。通过 OLAP 服务器，将这些数据抽取和转换为多维数据结构，以反映用户能理解的企业的真实维度。通过多维分析工具对信息的多个角度、多个侧面进行快速、一致和交互的存取，从而使分析员以及各层管理人员都能够对数据进行深入地分析和观察。

在数据仓库系统中 OLAP 使用的多维数据可以位于不同的层次，可以作为数据仓库的一部分，也可以作为数据仓库工具层的一部分。由于所处的层次的不同，其分析结果的综合程度也相应有高低之分，所以可以满足具有不同应用需求用户的要求。

2．难点分析

以下介绍的是 OLAP 的 12 条准则。

① OLAP 模型必须提供多维概念视图：从用户分析员的角度来看，整个企业的视图在本质上是多维的，因此 OLAP 的概念模型也应是多维的。企业决策分析的目的不同，从而决定了分析和衡量企业的数据总是从不同的角度来进行的，所以企业数据空间本身就是多维的。

② 透明性准则：无论 OLAP 是否是前端产品的一部分，对用户来说，它都是透明的。如果在客户/服务器结构中提供 OLAP 产品，那么对最终分析员来说，它同样也应透明。透明性原则包括两层含义，首先，OLAP 在体系结构中的位置对用户是透明的。OLAP 应处于一个真正的开放系统结构中，允许分析工具嵌入到分析人员指定的任何位置而不影响嵌入工具的性能，这对保持用户现有的效率，保证良好的性能至关重要。同时必须保证 OLAP 的嵌入不会引入和增加任何复杂性。其次，OLAP 的数据源对用户也是透明的。用户只须

使用熟悉的查询工具进行查询，而不必关心输入 OLAP 工具的数据是来自于同构还是异构的企业数据源。

③ 存取能力准则：OLAP 系统不仅能进行开放的存取，而且还提供高效的存取策略。OLAP 用户分析员不仅能在公共概念视图的基础上对关系数据库中的企业数据进行分析，而且在公共分析模型的基础上还可以对关系数据库、非关系数据库和外部存储的数据进行分析。OLAP 系统应提供高效的存取策略，应使系统只存取与指定分析有关的数据，避免多余的数据存取。

④ 稳定的报表性能：当数据的维数和综合层次增加时，提供给最终分析员的报表能力和响应速度不应该有明显的降低和减慢，这对维护 OLAP 产品的易用性和低复杂性至关重要。即便当企业模型改变时，关键数据的计算方法也无须更改。只有做到这一点，OLAP 工具提供的数据报表和所做的预测分析结果才是可信的。

⑤ 客户/服务器体系结构：OLAP 是建立在客户/服务器体系结构上的。这要求它的多维数据库服务器能够被不同的应用和工具访问到。服务器端智能地以最小的代价完成同多种服务器之间的挂接任务，服务器端必须完成分散的企业数据库的逻辑模式和物理模式之间的映射，并确保它们的一致性，从而保证透明性和建立统一的公共概念模式、逻辑模式和物理模式。客户端负责应用逻辑和用户界面。

⑥ 维的等同性准则：每一数据维在数据结构和操作能力上都是等同的。系统可以将附加的操作能力赋予给所选的维，但必须保证该操作能力可以赋予给其他任意的维，即要求维上的操作是公共的。

⑦ 动态的稀疏矩阵处理准则：OLAP 工具的物理模型必须充分适应指定的分析模型，提供“最优”的稀疏矩阵处理，这是 OLAP 工具所应遵循的最重要的准则之一。该准则包括两层含义，第一，对任意给定的稀疏矩阵，存在且仅存在一个最优的物理视图，它能提供最大的内存效率和矩阵处理能力。稀疏度是数据分布的一个特征，如果不能适应数据集合的数据分布，将会导致快速、高效操作的失败。第二，OLAP 工具的基本物理数据单元可配置给可能出现的维的子集。同时，还要提供动态可变的访问方法并包含多种存取机制，使得访问速度不会因数据维的多少、数据集的大小而变化。

⑧ 多用户支持能力准则：多个用户分析员可以同时工作于同一分析模型上，或者可以在同一企业数据上建立不同的分析模型，该准则可由准则 5 推出。OLAP 工具必须提供并发访问、数据完整性及安全性机制。实际上，OLAP 工具必须支持多用户也是为了适合数据分析工作的特点。推荐以工作组的形式来使用 OLAP 工具，这样多个用户可以交换各自的想法和分析结果。

⑨ 不受限的跨维操作：多维数据之间存在固有的层次关系，这就要求 OLAP 工具能自己推导出而不是由最终用户明确定义出相关的计算。对于无法从固有关系中得出的计算，要求系统提供计算完备的语言来定义各类计算公式。该准则是对准则 1 的一个补充，对操作能力和操作范围做出了要求。

⑩ 直观的数据操纵：要求数据操纵直观易懂。综合路径重定位、向上综合、向下挖掘和其他操作都可以通过直观、方便的单击操作完成。

灵活的报表生成：报表必须从各种可能的方面显示出从数据模型中综合出的数据和信息，充分反映数据分析模型的多维特征。

不受限制的维数与聚集层次：OLAP 工具的维数应不小于 15 维，用户分析员可以在任意给定的综合路径上建立任意多个聚集层次。

3．典型例题

【例题 8-11】 OLAP 是以____①____进行分析决策的基础，针对特定问题的联机数据访问和分析，OLAP 的数据来源于____①____。通过 OLAP 服务器，将这些数据抽取和转换为____②____，以反映用户能理解的企业的真实维度。

① A．数据库　　B．数据仓库　　C．数据文件　　D．数据格式

② A．数据文件　　B．数据模型　　C．数据格式　　D．多维数据结构

【答案】 ① B　② D

【解析】 OLAP 是以数据仓库进行分析决策的基础，针对特定问题的联机数据访问和分析，OLAP 能够对不同数据集合进行基于某个或是多个角度的比较，它能够从不同角度切割数据集合从而进行分析。

OLAP 的数据来源于数据库。通过 OLAP 服务器，将这些数据抽取和转换为多维数据结构，以反映用户能理解的企业的真实维度。通过多维分析工具对信息的多个角度、多个侧面进行快速、一致和交互的存取，从而使分析员和各层管理人员都能够对数据进行深入地分析和观察。

8.3.5 联机事务处理

联机事务处理系统（On-Line Transaction Processing，OLTP）是当今信息管理系统中应用极为普遍的处理方式，尤其在交融、证券、航空等对实时行要求比较高的领域。它要求操作系统具有多任务处理能力，优良的网络处理能力，硬件系统数据存储能力。

1．知识点提炼

（1）OLTP 概述

联机事务处理数据库应用程序最适合于管理变化的数据，通常这种应用程序有大量的用户同时执行更改实时数据的事务。尽管用户对数据的单个请求一般只引用少量记录，但是，这些请求有许多是同时发生的。这些类型的数据库的常见例子是航空订票系统和银行事务系统。在这种类型的应用程序中，主要关心的是并发性和原子性。

数据库系统中的并发性控制确保两个用户不能更改同一数据，或者一个用户不能在另一个用户对数据操作完成之前对其进行更改。例如，如果您正在与一位航空订票代理联系预订某航班上最后一个可用座位，该代理开始用您的姓名进行座位的预订处理，这时，其他代理应该不能再告诉其他乘客还可以预订该座位。

原子性确保事务中包括的所有步骤都作为一个组成功地完成。如果一个步骤失败，则不应完成其他步骤。例如，某个银行事务可能包括两个步骤：从您的支票账户中取出资金，然后将其放入您的存款账户中。如果从您的支票账户中成功地移走了资金，就需要确保将该资金放入您的存款账户中或重新放回到您的支票账户中。

（2）OLTP 特征

① 快速事务响应和频繁的数据修改

OLTP 以快速事务响应和频繁的数据修改为特征，用户利用数据库快速地处理具体业务。OLTP 应用时有频繁的写操作，所以数据库要提供数据锁、事务日志等机制。OLTP 应用要求多个查询并行，以便将每个查询的执行分布到一个处理器上。

② 事务量大

OLTP 的特点在于事务量大，但事务内容比较简单且重复率高。大量的数据操作主要涉及的是一些增加、删除、修改操作，一般仅仅涉及一张或几张表的少数记录，因此 OLTP 适合于处理高度结构化的信息。与其相适应，在数据组织方面 OLTP 以应用为核心、是应用驱动的，数据模型采用 E-R 模型。

（3）OLTP 设计注意事项

联机事务处理系统数据库应支持以下功能。

① 合理的数据存储

对于 OLTP 系统，输入/输出瓶颈是一个尤为关心的问题，原因在于修改整个数据库中数据的用户很多。确定数据的可能访问模式，并将经常访问的数据放在一起。在此过程中，可辅以文件组和 RAID（独立磁盘冗余阵列）系统。

② 减少长期锁

在事务期间，避免用户交互。无论何时，只要有可能，就通过执行单个存储过程来处理整个事务。在事务内对表的引用顺序可能会影响并发性，将对经常访问的表的引用置于事务的末尾，以便将控制锁的持续时间减至最短。

③ 联机备份

OLTP 系统通常的特征是连续操作（一天 24 小时，一周 7 天），为达到此目的，停工时间要保持绝对最短。尽管数据库可以在其正使用时进行备份，但是应将备份过程安排在活动不频繁时进行，以使对用户的影响减至最小。

④ 数据库的高度规范化

尽可能减少冗余信息以提高更新的速度，从而提高并发性。减少数据还可以提高备份的速度，因为只需要备份更少的数据。

⑤ 尽量减少历史数据

可以将很少引用的数据归档到单独的数据库中，或者从经常更新的表中移出，并置于仅含历史数据的表中。这将保持表尽可能地小，从而缩短备份时间，改善查询性能。

⑥ 谨慎使用索引

每次添加或修改行时，必须更新索引。若要避免对经常更新的表进行过多的索引，索引范围应保持较窄。请用索引优化向导设计索引。

2．难点分析

下面介绍一下 OLTP 与 OLAP 之间的关系。

OLTP 处理的数据是高度结构化的，涉及的事务比较简单，因此复杂的表关联不会严重影响性能。反之，决策支持系统的一个查询可能涉及数万条记录。这时复杂的联接操作会严重影响性能。在 OLTP 系统中，数据访问路径是已知的，至少是相对固定的，应用程序可以在事务中使用具体的数据结构如表、索引等。而决策支持系统使用的数据不仅有结构化数据，而且有非结构化数据，用户常常是在想要某种数据前才决定去分析该数据。因此数据仓库系统中一定要为用户设计出更为简明的数据分析模型，这样才能为决策支持提供更为透明的数据访问。

OLAP 是以数据仓库为基础的，其最终数据来源与 OLTP 一样均来自底层的数据库系统，但由于二者面对的用户不同：OLTP 面对的是操作人员和低层管理人员，OLTP 面对的是决策人员和高层管理人员，因而数据特点与数据处理方式也明显不同。OLTP 和 OLAP 的区别如图 8-22 所示。

OLTP	OLAP
数据库原始数据	数据库导出数据或数据仓库数据
细节性数据	综合性数据
当前数据	历史数据
经常更新	不可更新，但周期性刷新
一次性处理的数据量小	一次性处理的数据量大
响应时间要求高	响应时间合理
用户数量大	用户数量相对较少
面向操作人员，支持日常操作	面向决策人员，支持管理需要
面向应用，事务驱动	面向分析，分析驱动

图 8-22　OLTP 与 OLAP 对比表

由图 8-22 可见，OLTP 与 OLAP 是两类不同的应用。OLTP 面对的是操作人员和低层管理人员，OLAP 面对则是决策人员和高层管理人员；OLTP 是对基本数据的查询和增加、删除、修改操作处理，它以数据库为基础，而 OLAP 更适合以数据仓库为基础的数据分析处理。OLAP 所需的历史的、导出的及经综合提炼的数据均来自 OLTP 所依赖的底层数据库。OLAP 数据较之 OLTP 数据而言要增加了数据多维化或预综合处理等操作。例如，对一些统计数据，首先进行预综合处理，建立不同层次级别的统计数据，从而满足快速统计分析和查询的要求。除了数据及处理上的不同之外，OLAP 的前端产品的界面风格及数据访问方式也同 OLTP 有所区别。OLTP 多为操作人员经常用到的固定表格，查询和数据显示也比较固定、规范。而 OLAP 多采用便于非数据处理专业人员理解的方式，如多维报表、

统计图形等，查询及数据输出直观灵活，用户可以方便地进行逐层细化、切片、切块和数据旋转等操作。

3．典型例题

【例题 8-12】 联机事务处理系统，是当今信息管理系统中应用极为普遍的处理方式，最适合于管理＿①＿的数据。在这种类型的应用程序中，主要关心的是＿②＿和原子性。

① A．静态　　B．变化　　C．容量增长　　D．无固定格式的

② A．数据分布状态　　B．数据模型　　C．并发性　　D．数据结构

【答案】 ① A　② C

【解析】 联机事务处理数据库应用程序最适合于管理变化的数据，通常这种应用程序有大量的用户同时执行更改实时数据的事务。尽管用户对数据的单个请求一般只引用少量记录，但是，这些请求有许多是同时发生的。这些类型的数据库的常见例子是航空订票系统和银行事务系统。在这种类型的应用程序中，主要关心的是并发性和原子性。

数据库系统中的并发性控制确保两个用户不能更改同一数据，或者一个用户不能在另一个用户对数据操作完成之前对其进行更改。例如，如果您正在与一位航空订票代理联系预订某航班上最后一个可用座位，该代理开始用您的姓名进行座位的预订处理，这时，其他代理应该不能再告诉其他乘客还可以预订该座位。

原子性确保事务中包括的所有步骤都作为一个组成功地完成。如果一个步骤失败，则不应完成其他步骤。例如，某个银行事务可能包括两个步骤：从您的支票账户中取出资金，然后将其放入您的存款账户中。如果从您的支票账户中成功地移走了资金，就需要确保将该资金放入您的存款账户中或重新放回到您的支票账户中。

练习题

试题一

1．试用 ORDB 的定义语言，定义符合下列条件的数据库。

2．试用 ORDB 的查询语言，分别写出下列查询的 SELECT 语句。

① 检索“计算机系”每一个学生的学习成绩，要求显示（Sno，Sname，Cname，Grade）。

② 检索选修本系课程的学生选课情况，要求显示（Dno，Dname，Sno，Sname，Cno，Cname）。

③ 检索选修本系课程、并由本系教师任教的课程的学生选课情况，要求显示（Dno，Dname，Sno，Sname，Cno，Cname，Fno，Fname）。

④ 检索“计算机系”每一个男学生选修课程的门数，要求显示（Sno，Sname，Course_Num）。

【说明】

以下是某大学内关于系（Dept）、学生（Student）、成绩（SC）、课程（Course）和教

师（Faculty）的信息说明。

Dept 是有关学校里系信息的对象类型，有 6 个属性。两个是基本数据类型，系编号（Dno）和系名（Dname）；单值属性 Director 表示有一位教师是系主任；还有 3 个是多值属性，Staff 表示系里有若干教师，Mass 表示系里有若干学生，Set_Up 表示系里开设了若干门课程。

Student 是有关学生信息的对象类型，有 6 个属性。4 个是基本数据类型，学生的学号（Sno）、姓名（Sname）、年龄（Age）和性别（Sex）；单值属性 study_In 表示学生属于某个系；多值属性 study 表示该学生的学习成绩。

SC 是有关成绩信息的对象类型，有 3 个属性。成绩 Grade 是基本数据类型；单值属性 Student 表示该成绩是属于何学生；单值属性 Course 表示该成绩属于哪门课程。

Course 是有关课程信息的对象类型，有 5 个属性。两个是基本数据类型，课程号（Cno）和课程名（Cname）；有两个是单值属性，Teacher 表示课程的任课老师，Founder 表示课程由何系设置的；多值属性 Learn 表示选修这门课程的学生成绩。

Faculty 是有关教师信息的对象类型，有 5 个属性。3 个是基本数据类型，教师工号（Fno）、姓名（Fname）和工资（Salary）；单值属性 Works_For 表示教师服务的系；多值属性 Teach 表示教师开设了若干门课程。

试题二

1．试用 ORDB 的定义语言，定义这个数据库。

2．试用 ORDB 的查询语言，分别写出下列查询的 SELECT 语句。

① 检索每个学生的学习课程和成绩。

② 检索至少有一门课程的求学地与籍贯在同一城市的学生学号和姓名。

【说明】

对象 student 包含身份证号、姓名、籍贯和学习（Studies）等属性，对象 Study 包含课程名、成绩、求学地、大学以及学生（Student）等属性。对象 Student 和 Study 之间联系为 1∶N。

试题三

阅读下列说明和有关的图，回答问题 1 到问题 4，将解答填入答题纸的对应栏内。

【说明】

某制造企业的 ERP 物料出入库管理的工作流程分别叙述如下。

（1）出库工作流程

① 领料人提交领料单（每一种物料有一张领料单）；

② 仓库保管员根据领料计划单检验该领料单是否有效；

③ 若经检验没有相应的领料计划，则通知领料人该领料单无效；

④ 若领料单有效，仓库保管员根据领料单上的物料代码核对是否有足够的库存；

⑤ 若没有足够的库存，仓库保管员向领料人发缺货单；

⑥ 若有足够的库存，仓库保管员在领料单上签字，并登记出库单，修改物料主文件中的现有库存数；相应的物料出库，物料清单交领料人。

（2）入库工作流程

① 采购员提交入库申请单（每一种物料有一张入库申请单）；

② 仓库保管员根据采购计划单验收入库申请单；

③ 若验收发现没有相应的采购计划，则仓库保管员向采购员发无效申请单；

④ 若验收合格，则仓库保管员向检验员申请物料检验；检验员根据检验结果填写物料检验单；

⑤ 如果物料或供货方不合格，则向采购员发出退货单；

⑥ 如果检验合格，则仓库保管员登记入库单，修改物料主文件中的现有库存数，相应的物料入库。

为便于及时了解库存情况、核查出入库情况，该企业决定将上述人工流程纳入 ERP 系统设计范围内。在设计该系统时，采用了两种方法：结构化方法和面向对象方法。

图 8-23 给出了物料出入库系统的数据流图，图中的数据流并没有画全，需要考生填补。图 8-24 给出了采用面向对象方法所认定出的类。

图 8-23　物料出入库系统的数据流图

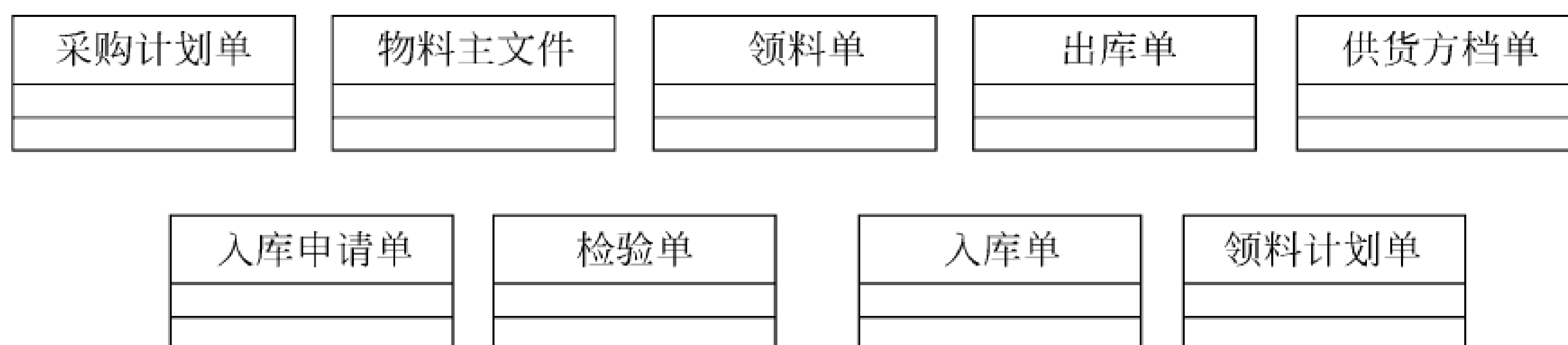


图 8-24　物料出入库系统中的类

【问题 1】

图 8-23 中缺少了哪些数据流?请指明每条数据流的名称、起点和终点。

【问题 2】

给出“领料单”和“入库申请单”这两个类至少应具有的属性。

【问题 3】

为建立功能完善的 ERP 库存管理系统，除了查询、统计、报表输出功能外，还应具有哪些对提高企业效益至关重要的功能?

【问题 4】

用面向对象方法设计的类中，有一些类的对象是需要持久存储的，这样的类一般需要映射到关系数据库模式中。请指出图 8-24 中哪些类需要做这样的映射。

练习题答案

试题一

1．解：符合条件的 ORDB 的定义语言如下

```
        CREATE  TYPE  MyString  CHAR  Varying;
        CREATE  TABLE  DEPT(Dno      MyString,
 Dname  MyString,
 Mass       SETOF(ref(student)),
                              Set_Up    SETOF(Ref(course)),
                              DIRECTOR  REF(Faculty),
                              Staff     SETOF(REF(Faculty)));
 CREATE  TABLE  Student(Sno      MyString,
     Sname       MyString,
   Age           INTEGER,
                                Sex         MyString,
                                Study_In    ref(dept),
                                Study       SETOF(REF(Sc)));
```

```
CREATE  TABLE  SC(Grade      INTEGER,
Student      REF(Student),
                      Course     REF(Course));
CREATE  TABLE  Course(Cno        MyString,
   Cname          MyString,
   Founder        REF(Dept),
                                 teacher     ref(faculty));
CREATE  TABLE  Faculty(Fno       MyString,
   Fname          MyString,
   Salary         INTEGER,
                                 Works_For   REF(Dept),
                                 Teach       SETOF(REF(Course)));
```

2．解：

```
① SELECT  A.Sno, A.Sname, B.Course.Cname, B.Grade
   FROM  Student  As  A, A.Study  AS  B
   WHERE  A.Study_In.Dname='计算机系';
或者    SELECT  A.Sno, A.Sname, B.Course.cname, B.Grade
   FROM  Dept  AS  D, D.Mass  AS  A, A.Study  AS  B
   WHERE  D.Dname='计算机系';
② SELECT  A.Study_In.Dno, A.Study_In.Dname, A.Sno, A.Sname, B.Course.Cno,
   B.Course.Cname
   FROM  Student  AS  A, A.Study  AS  B
   WHERE  A.Study_In.Dno=B.Course.Founder.Dno;
或者    SELECT  D.Dno, D.Dname, A.Sno, A.Sname, B.Course.Cno, B.Course.Cname
   FROM  Dept  AS  D, D.Mass  AS  A, A.Study  AS  B
   WHERE  D.Dno=B.Course.Founder.Dno;
③ SELECT  A.Study_In.Dno, A.Study_In.Dname, A.Sno, A.Sname, B.Course.Cno,
   B.course.cname, B.Course.Teacher.Fno, B.Course.Teacher.Fname
   FROM  Student  AS  A, A.Study  AS  B
   WHERE  A.Study_In.Dno=B.Course.Founder.Dno
AND  A.Study_In.Dno=B.Course.Teach.Works_For.Dno;
④ SELECT  A.Sno, A.Sname, COUNT(SELECT  *
FROM  A.Study)
   FROM  Student  AS  A
   WHERE  A.Sex='M';
```

试题二

```
（1）CREATE  TYPE  MyString  CHAR  Varying;
     CREATE  TABLE  Student(Sno  INTEGER,
```

```
                                Sname  MyString,
                                City  MyString,
                                Studies  SETOFF(REF(Study)));
CREATE  TABLE  Study(Coursename  MyString,
                                Grade  Integer,
                                City  MyString,
                                University  MyString,
                                Student  REF(Student));
```

（2）

```
① SELECT  A.Sname,  B.Coursename,  B.Grade
  FROM  Student  AS  A,  A.Studies  AS  B;
② SELECT  A.Sno,  A.Sname
        FROM  Student  AS  A,  A.Student  AS  B
        WHERE  A.City=B.City;
```

试题三

（1）名称：退货单；起点：物料检验；终点：采购员。

名称：缺货单；起点：领料单检验；终点：领料人。

（2）领料单的属性：物料代码、数量、日期、领料人、仓库保管员。

入库申请单的属性：物料代码、数量、供货方、日期、采购员。

（3）库存超限报警、库存不足报警。

（4）采购计划单、入库单、供货方档案、出库单、物料主文件、领料计划单。